НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Алгоритм і програма формування послідовності рівновагових двійкових векторів

Виконала: студентка IV курсу, групи КВ-51

Пашкова Анжеліка Миколаївна _ _____
(підпис)

Керівник проф.каф.СПіСКС, д.т.н., доцент Романкевич В.О._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПіСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ. 045490.001 ОА	Опис альбому	2	
3	А4	ІАЛЦ. 045490.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ. 045490.003 ТП	Відомість технічного проекту	2	
5	А4	ІАЛЦ. 045490.004 ПЗ	Пояснювальна записка	50	
6	А4	ІАЛЦ. 045490.005 Д1	Діаграма класів. Структурна схема	1	
7	А4	ІАЛЦ. 045490.006 Д2	Перевірка функції. Схема алгоритму	1	
8	А4	ІАЛЦ. 045490.007 Д3	Пошук функцій. Схема алгоритму	1	
9	А4	ІАЛЦ. 045490.008 Д4	Форматування виводу функції. Схема алгоритму	1	
10		Додатки		2	
11		Диск CD-ROM	Матеріали дипломного проекту	1	

				ІАЛЦ. 045490.003 ТП		
	ПІБ	Підп.	Дата			
Розробн.	Пашкова			Відомість технічного проекту	Лист	Листів
Керівн.	Романкевич				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-51	
Н/контр.	Клятченко					
Зав.каф.	Тарасенко					

Пояснювальна записка
до дипломного проекту

на тему: Алгоритм і програма формування

послідовності рівновагових двійкових векторів

Київ – 2019 року

Додаток 1

Копії графічного матеріалу

Київ – 2019 року

Додаток 2

Фрагменти програмного коду

Київ – 2019 року

Додаток 3

Слайди преезентації

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студентки
Пашкової Анжеліки Миколаївни
(прізвище, ім’я, по батькові)

1. Тема проекту : Алгоритм і програма формування послідовності рівновагових двійкових векторів, керівник проекту доц.каф.СПСКС, д.т.н. Романкевич В.О., затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту : див. Технічне завдання

4. Зміст пояснювальної записки:
- аналіз існуючих рішень та обґрунтування теми дипломного проекту;
- аналіз способів та технологій розробки;
- програмне забезпечення для формування послідовностей рівновагових двійкових векторів.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) :
- Діаграма класів. Структурна схема;
- Перевірка функції. Схема алгоритму;
- Пошук функцій. Схема алгоритму;
- Форматування виводу функції. Схема алгоритму;
- Слайди презентації.

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	доц.каф.СПСКС, к.т.н. Клятченко Я.М.	05.11.2018	

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення літератури за тематикою проекту	19.11.2018	
2	Розроблення та узгодження технічного завдання	04.12.2019	
3	Аналіз існуючих рішень	21.01.2019	
4	Підготовка матеріалів першого розділу дипломного проекту	11.02.2019	
5	Підготовка матеріалів другого розділу дипломного проекту	18.03.2019	
6	Підготовка графічної частини дипломного проекту	01.04.2019	
7	Оформлення документації дипломного проекту	13.05.2019	
8	Попередній огляд матеріалів диплому на кафедрі	30.05.2019	

Студент _____ Пашкова А.М.
(підпис)

Керівник проекту _____ Романкевич В.О.

* Консультантом не може бути зазначено керівника дипломного проекту.

АНОТАЦІЯ

Кваліфікаційна робота включає в себе пояснювальну записку (50 стор., 13 рис.).

Об’єктом розробки є створення комп’ютерної програми для формування послідовності рівновагових двійкових векторів за допомогою генератора на базі регістру зсуву зі зворотнім зв’язком.

Дане програмне забезпечення дозволяє для заданих довжини та ваги векторів знайти різні функції управління регістром зсуву, а також отримати послідовність двійкових векторів, сформовану за допомогою згенерованої функції.

В ході розробки:

- проведено аналіз роботи алгоритмів формування послідовностей псевдовипадкових чисел;
- проаналізовано принцип роботи та структуру генератора, що базується на регістрі зсуву з лінійним зворотнім зв’язком;
- проаналізовані мови програмування, за допомогою яких можлива реалізація потрібного програмного забезпечення;
- спроектовано структуру генератора на базі регістру зсуву зі зворотнім зв’язком;
- розроблено структуру взаємодії програмних модулів генератора;
- розроблено програмне забезпечення для управління характеристиками з генератором.

Застосування даного програмного забезпечення дозволить знаходити функції управління генератором і сформувати послідовності спеціалізованих рівновагових двійкових векторів, що використовуються, наприклад, під час розрахунку показників надійності відмовостійких багатопроцесорних систем.

Ключові слова:

Генератор псевдовипадкових двійкових векторів, розрахунок показників надійності, відмовостійкі багатопроцесорні системи, функції управління генератором.

ANOTATION

The qualifying work includes an explanatory note (50 page, 13 fig.).

The creation of a computer program for forming a sequence of equilibrium binary vectors using a generator on the basis of the shift register with feedback is the object of this development.

This software allows for a given vector length and weight to find different functions of control of the shift register, as well as to obtain a sequence of binary vectors generated by the generated function.

During development:

- the algorithm analysis of forming pseudorandom numbers sequences has been made;

- the operation principle and structure of the generator based on the shift register with linear feedback has been analyzed;

- programming languages that make the implementation of necessary software possible have been analyzed;

- generator structure has been designed on the basis of the shift register with feedback;

- program modules interaction structure of the generator has been designed;

- software for managing the characteristics of the generator has been developed.

The use of this software will enable the management of generator functions and determine sequencing of specialized equilibrium binary vectors used, for example, when calculating the reliability indicators of fault-tolerant multiprocessor systems.

Keywords:

Generator of pseudorandom binary vectors, calculation of reliability indicators, fault-tolerant multiprocessor systems, generator control functions.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
1	А4	ІАЛЦ. 045490.002 ТЗ	Алгоритм і програма	4		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Технічне завдання			
2	А4	ІАЛЦ. 045490.003 ТП	Алгоритм і програма	2		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Відомість технічного			
			проекту			
3	А4	ІАЛЦ. 045490.004 ПЗ	Алгоритм і програма	53		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Пояснювальна записка			
4	А4	ІАЛЦ. 045490.005 Д1	Алгоритм і програма	1		
			формування послідовності			
			рівновагових двійкових			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 045490.001 ОА		
Розробив		Пашкова А.М.			Алгоритм і програма формування послідовності рівновагових двійкових векторів Опис альбому	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Романкевич В.О.				КПІ ім. Ігоря Сікорського ФПМ КВ-51	
Н. контроль		Клятченко Я.М.					
Затвердив		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			векторів.			
			Діаграма класів.			
			Структурна схема			
5	А4	ІАЛЦ. 045490.006 Д2	Алгоритм і програма	1		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Перевірка функції.			
			Схема алгоритму			
6	А4	ІАЛЦ. 045490.007 Д3	Алгоритм і програма	1		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Пошук функцій.			
			Схема алгоритму			
7	А4	ІАЛЦ. 045490.008 Д4	Алгоритм і програма	1		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Форматування виводу			
			Функції.			
			Схема алгоритму			
8		Додатки				

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
9	А4	CD-ROM	Матеріали дипломного проекту	1		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 045490.001 ОА	Арк. 3

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Алгоритм та програма генрації рівновагових двійкових веторів».

Галузь застосування: імітаційне моделювання, разрахування показників надійності відмовостійких багатопроцесорних систем.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення прикладного програмного забезпечення для пошуку функцій затримки регістру зсуву з лінійним зворотнім зв'язком і формування послідовності рівновагових двійкових векторів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет, в яких порушуються ці питання.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з операційними системами Windows, Linux;

- можливість управління характеристиками генератора;
- можливість вибору логічних операцій для функцій управління генератором;
- можливість вибору функції управління генератором для формування послідовності рівновагових двійкових векторів;
- можливість запису згенерованої послідовності у файл;
- наявність зручного інтерфейсу користувача.

5.2. Вимоги до апаратного забезпечення

- Процесор: 2-ядерний, 500 МГц;
- Оперативна пам’ять: 256 Мб;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Linux;
- Підтримка OpenGL ES 2.0;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	19.11.2018
2.	Розроблення та узгодження технічного завдання	04.12.2019
3.	Аналіз існуючих рішень	21.01.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	11.02.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.03.2019
6.	Підготовка графічної частини дипломного проекту	01.04.2019
7.	Оформлення документації дипломного проекту	13.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
1	А4	ІАЛЦ. 045490.004 ПЗ	Алгоритм і програма	53		
			формування послідовності			
			рівновагових двійкових			
			векторів.			
			Пояснювальна записка			
2	А4	ІАЛЦ. 045490.005 Д1	Діаграма класів.	1		
			Структурна схема			
3	А4	ІАЛЦ. 045490.006 Д2	Перевірка функції.	1		
			Схема алгоритму			
4	А4	ІАЛЦ. 045490.007 Д3	Пошук функцій.	1		
			Схема алгоритму			
5	А4	ІАЛЦ. 045490.008 Д4	Форматування виводу	1		
			функції.			
			Схема алгоритму			
6		Додатки				
7	А4	CD-ROM	Матеріали проекту	1		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 045490.003 ОА		
Розробив		Пашкова А.М.			Алгоритм і програма формування послідовності рівновагових двійкових векторів Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Романкевич В.О.					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського ФПМ КВ-51	
Затвердив		Тарасенко В.П.					

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БОС – багатопроцесорні обчислювальні системи;

БФ – булева функція;

ВБС – відмовостійкі багатопроцесорні системи;

ВЧ – випадкові числа;

ГВЧ – генератор випадкових чисел;

ГПЧ – генератор псевдовипадкових чисел;

ООП – об'єктно-орієнтоване програмування;

ПЗ – програмне забезпечення;

ПП – псевдовипадкові послідовності;

ПЧ – псевдовипадкові числа;

РДВ – рівновагові двійкові вектори;

ФУ – функція управління;

ВСТУП

В наш час відмовостійкі багатопроцесорні системи (ВБС) є невід'ємною частиною розвитку комп'ютерних технологій. Використовуються в космічних, авіаційних, атомних та інших системах управління. Область застосування ВБС є досить великою і складність таких систем постійно збільшується. Вони потребують особливої уваги щодо розрахунку надійності, так як вихід з ладу такої системи може призвести до катастрофічних наслідків.

Під час проектування ВБС важливим є етап розрахунку надійності. Серед відомих методів оцінки надійності обчислювальних систем є детерміновані та ймовірнісні методи. В складних спеціалізованих системах більшого поширення знаходять ймовірнісні методи, серед яких велика увага приділяється розвитку методів псевдовипадкового тестування. Для цього необхідне моделювання таких систем в потоці відмов.

Основою методів розрахунку надійності є графо-логічні моделі, експерименти з якими дають можливість отримання оцінки параметрів системи. Такі моделі визначають реакцію системи на появу певного двійкового вектора, в якому 0 вказує на несправний елемент, а 1 – на справний. Для формування таких наборів векторів потрібні генератори псевдовипадкових чисел (ГПЧ).

З'явився навіть новий розділ математики, що вивчає властивості ГПЧ для використання в криптографії. Псевдовипадкові числа (ПЧ) потрібні для алгоритмів шифрування, генерації унікальних ідентифікаторів, унікальних ключів шифрування. В будь-якій системі передачі секретних даних потрібна велика кількість ключів для всіх користувачів системи, яку можна отримати за допомогою ГПЧ. Також ПЧ є потрібними для потокових шифрів. Результат роботи таких генераторів має не відрізнятись від випадкових послідовностей чисел, а з іншої сторони бути повністю передбачуваним. Є різні методи побудови таких генераторів. Вони задовольняють умовам лінійної складності,

великого періоду, рівномірного розподілу. Але вказані генератори, на жаль, не мають властивостей, які корисні для використання під час досліджень у ВБС, так як не гарантують отримання всіх можливих векторів послідовності без повторень. Такі генератори потребують спеціальних умов.

Завданням дипломного проекту є формування послідовності двійкових векторів за допомогою генератора, побудованого на базі регістру зсуву з прямим зворотнім зв’язком. Особливістю роботи такого генератора є управління окремими розрядами регістра під час зсуву та формування всіх можливих рівновагових векторів почергово без повторів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Різновиди генераторів послідовностей чисел

Випадкові числа (ВЧ) знаходять використання в криптографії, різних додатках, рішенні проблеми моделювання багатопроцесорних систем. Основними критеріями оцінки є випадковість та непередбачуваність послідовностей таких чисел.

При створені послідовності ВЧ вважається, що дана послідовність чисел має бути випадковою у статистичному сенсі. Для цього є 2 критерії: однорідний розподіл і незалежність.

Незалежність означає що жодне значення в послідовності не має залежати від інших. Розподіл чисел має бути однорідним, що означає, що частота появи всіх утворених чисел має бути однаковою. Можливе проведення експериментів, що підтверджують однорідний розподіл випадкової величини та підтверджують залежність послідовності, але немає експериментів, які покажуть що послідовність є незалежною. При дійсно випадковій послідовності кожне наступне число статистично не має залежності від інших чисел, тобто є непередбачуваними.

Розрахунок надійності безвідмовної роботи ВБС управління складними об’єктами на етапі розробки й проектування досить часто відбувається за рахунок проведення експериментів із математичними моделями, що коректно відображають реакцію такої відмовостійкої системи на появу відмов одного чи декількох модулів системи. Зазвичай, можливості провести експерименти з моделлю над всіма можливими векторами стану немає, так як кількість модулів є великою, що вимагає великі затрати часу та формування дуже великою кількості векторів.

За Хеммінгом завжди можна підібрати два значення, такі як мінімальна кількість k_{min} та максимальна кількість модулів k_{max}, що при значеннях строго

менше чи більше відповідно, система залишається працездатною чи відмовляє відповідно. Це надає можливість зменшити необхідну кількість експериментів. Ймовірність знаходження системи у множині станів, описаних двійковими векторами з вагою, що не знаходиться в діапазоні знайдених мінімального k_{min} та максимального k_{max} значень, легко обчислюється відомими алгоритмами.

Будь-який ГПЧ з обмеженими ресурсами в певний момент часу зациклюється, тобто починає повторювати необмежену кількість разів однакову числову послідовність. Довжина цих циклів залежить від структура самого генератора і в середньому величина циклів приблизно $2^{n/2}$, де n – розмір внутрішніх станів у бітах. Лінійні конгруентні генератори та генератори на базі регістрів зсуву мають максимальні періоди приблизно 2^n. Якщо цикл сформованої генератором послідовності є надто коротким, то такий ГПЧ є передбачуваним і стає непридатним до використання в ролі генератора випадкових чисел (ГВЧ) [1].

Для генерації рівномірно розподілених випадкових величин використовуються:

- табличні генератори;
- фізичні генератори випадкових чисел;
- алгоритмічні генератори.

Результатом роботи алгоритмічних генераторів є ПЧ, так як формуються за допомогою заданої формули. Ці методи є більш простими у реалізації на ЕОМ, тому є найбільш популярними. Серед них виділяють конгруентні методи та методи, що базуються на використанні регістру зсуву з лінійним зворотнім зв’язком.

Табличні ГВЧ в якості джерела використовують сформовані певним чином таблиці, які містять в собі перевірені незалежні між собою числа. При обході таблиці можна отримувати рівномірно розподілені випадкові числа. Недоліками даного методу є:

- для зберігання великої кількості цифр потрібно багато пам’яті;
- складність створення таких таблиць;
- перевірка створених таблиць;
- повторне використання такої таблиці вже не гарантує випадковості чисел.

Фізичні генератори утворюють послідовність ВЧ на базі вимірювань параметрів деякого фізичного процесу. Принцип дії заключається в перетворенні випадкового сигналу на виході з фізичного джерела шуму в послідовність імпульсів з ймовірністю появи високого рівня 0,5. Прикладом такого генератора можуть слугувати монета, гральні кубики, апаратний генератор шуму.

Видами фізичного генератора є:

- ГВЧ на основі використання випадкових станів бістабільної схеми при періодичному ввімкненні та вимкнені джерела струму;
- ГВЧ з перерахунком імпульсів періодичної послідовності за випадковий інтервал часу;
- ГВЧ з перерахунком імпульсів випадкової послідовності протягом фіксованого проміжку часу;
- ГВЧ на основі дискретизації неперервного шумового сигналу по рівням.

Найбільш часто для розробки швидкодіючих та якісних ГВЧ застосовуються два останні методи. Але існують і генератори, що використовують такі фізичні процеси як радіоактивний розпад, шум аналогових мереж, космічні випромінювання.

В генераторах з перерахунком імпульсів за фіксований інтервал часу однакова ймовірність символів вихідної послідовності досягається шляхом виконання операції суми за модулем 2, що знижує швидкодію генератора і обмежує сферу їх використання. Кращу швидкодію мають ГВЧ на базі

дискретизації неперервного шуму по двом рівням, але таким пристроям властива чутливість до змін первинного шумового сигналу.

Загальними недоліками фізичних генераторів є

- обмеженість швидкодії, що визначається первинним аналоговим джерелом шуму;
- низька стабільність основних ймовірнісних характеристик через нестабільність первинних джерел і джерел струму, що вимагає періодичної статистичної перевірки якості послідовності, що генерується;
- складність апаратної реалізації, викликана наявністю декількох джерел струму і необхідністю стабілізації та фільтрації їх напруги; неможливість відтворення і передбачення послідовностей, що генеруються, через випадкову природу їх утворення;
- неоднорідність структури через наявність і аналогових, і цифрових вузлів.

Числа, що генеруються за допомогою алгоритмічного методу є завжди псевдовипадковими [3]. Кожне наступне число залежить від попереднього. Такі послідовності утворюють цикли, які повторюються нескінченну кількість разів.

До таких методів відносяться:

- метод серединних квадратів;
- метод серединних добутків;
- метод перемішування;
- лінійний конгруентний метод.

Штучне збільшення періоду повтору псевдовипадкового сигналу необмежено наближує структуру генератора до структури одного з можливих реалізацій дійсно випадкового процесу. Миттєві значення таких ПП, на відміну від випадкових, можуть бути передбачені завчасно. Всі оцінки

статистичних характеристик конкретної реалізації ПП є такими ж, як і оцінки випадкової вибірки.

Але в певних умовах ПП можуть замінити випадкові. Під час аналізу частини ПП, кількість чисел в якій є рівною або меншою за довжину періоду, практично неможливо визначити, чи являється відрізком регулярної послідовності.

З точки зору реальних характеристик важко встановити різницю між випадковими та псевдовипадковими числовими послідовностями. В той же час ГПЧ має ряд переваг:

- періодичний характер ПЧ обумовлює низький рівень дисперсії оцінок, отриманих визначенні середнього значень цілого числа періодів;
- характеристики ГПЧ є стабільними та визначаються алгоритмом формування ПЧ;
- також послідовність можна повторювати з будь-якого потрібного моменту, для чого не вимагається складних пристроїв запам’ятовування.

Метод серединних квадратів заключається в тому, що є деяке число, яке підносять до квадрату та виділяють середні цифри, які й вважають наступний числом послідовності. Наприклад, є чотирьохзначне число 5678. Його підносять до квадрату, в результаті чого отримують восьмизначне число 32239684. Новим числом послідовності є 4 цифри, що знаходяться посередині утвореного, які на рис. 1 виділені прямокутником – 2396.

Недоліком даного генератора є те, що на деякій ітерації роботи значення може стати рівним 0 (наприклад, $5000^2 = 25000000$, тобто наступний числом є 0), що є виродженим випадком, так як під час наступних ітерацій стан регістру не змінюватиметься, так як результатом буде 0.

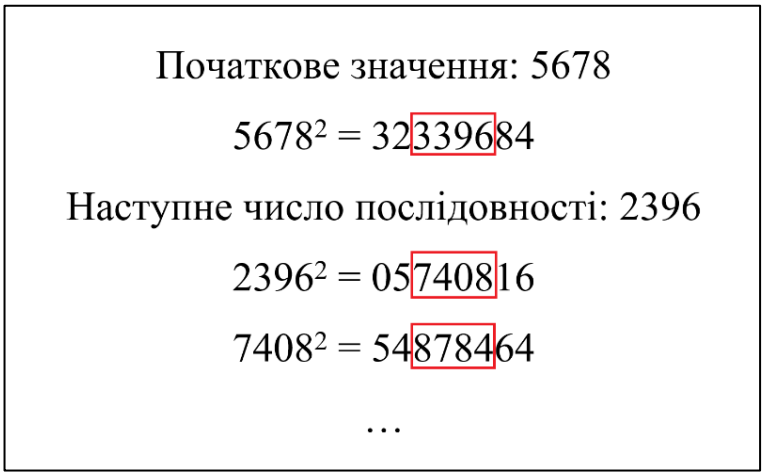

Рисунок 1 – Ілюстрація роботи методу серединних квадратів

У методі серединних добутків для початку роботи потрібні 2 числа. Розраховується третє число як добуток першого і другого чисел. Після цього з отриманого результату (третього числа) беруть середину і множать на друге число. За допомогою чого отримують четверте число, середину якого множитимуть на третє число в наступній ітерації. За потреби кроки повторюються.

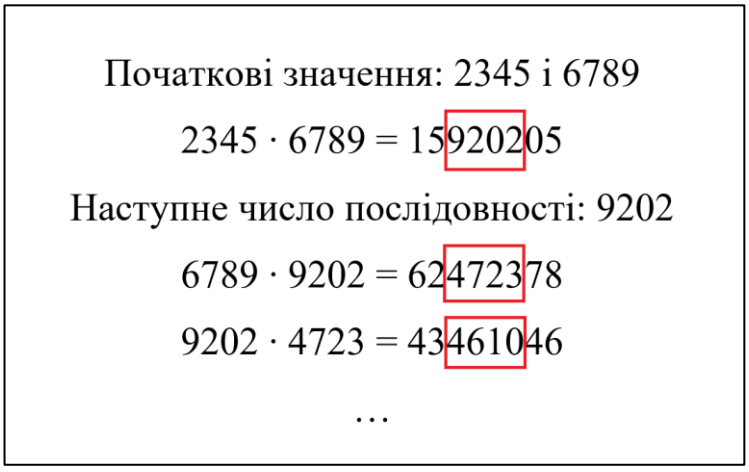

Рисунок 2 – Ілюстрація роботи методу серединних добутків

Наприклад, є два чотирьохзначні числа: 2345 і 6789. Їх добуток дорівнюватиме 15920205. Тобто наступним числом є 9202. Роботу даного прикладу показано на рис. 2. Для отримання чергового члена послідовності рахуємо добуток чисел 6798 і 9202, який дорівнюватиме 62555196, і беремо середні цифри. Буде отримано 5551. Вказані дії повторюються потрібну

кількість разів. Недоліком такого генератора є обмеженість, адже після певної ітерації буде отримане число 0.

Метод перемішування заключається у використанні зсувів цифр числа між собою. Початкове число зсувають вліво на певну кількість розрядів, отримуючи перший результат, і вправо – другий результат. Після цього добуток двох отриманих чисел і буде наступним у послідовності. У випадку, коли добуток має розрядність більшу за почтову, лишні розряди відкидаються. В деяких мовах програмування це відбувається автоматично.

Метод RSA складається з багатьох кроків. Спочатку генеруються 2 різних досить великих простих числа p, q та обчислюються числа N і φ:

$$N = pq;$$

$$\varphi = (p-1)(q-1);$$

Далі обирається число k, яке більше , менше φ, є взаємнопростим з φ так, щоб найбільшим спільним дільником чисел k і φ була 1. Другим кроком є вибір випадкового цілого додатнього числа u_0, яке буде менше N і гратиме роль стартового. Далі для $i = 1,2,\ldots,n$ розраховуються число u_i за формулою:

$$u_i = u_{i-1}^k mod\ N\ \in \{0,1,\ldots,N-1\}.$$

Також для кожного i розраховується x_i, що відповідає наймолодшому біту числа u_i у двійковому представленні.

$$x_i \in A = \{0,1\}.$$

Далі формується результуюча послідовність $x_1, x_2, \ldots, x_n$.

Недоліком RSA-алгоритму є його повільна швидкодія при реалізації на комп’ютерах універсального призначення, яка викликається великими затратами машинного часу для виконання множення за модулем при обчисленні u_i.

При формуванні послідовностей ПЧ за допомогою спеціальних алгоритмів є важливими умови: отримувана послідовність чисел повинна мати статичну структуру, що є достатньо близькою до рівномірно розподіленої, та

кількість операцій, що необхідна для формування кожного числа послідовності, має бути не надто великою.

Ще одним з варіантів є дробові генератори ПЧ. Але вони так само не застосовуються дуже широко. На практиці результатами їх роботи практично не користуються, тому що для забезпечення коректної роботи потрібно зберігати в пам'яті ірраціональні числа з нескінченною кількістю знаків після крапки, що є нереальним. А при заміні раціональними числами ймовірність отримати малий період циклу є занадто великою.

В апаратурних формувачах ПЧ і вузлах ЕОМ часто використовується метод, який заключається в отриманні лінійної двійкової послідовності за рекурентним виразом:

$$a_i = \sum_{k-1}^{m} \oplus a_k a_{i-k}, i = 0,1,2,3,\ldots,$$

Де i – номер такту, a_i – символ послідовності, що може приймати значення 0 та 1, a_k – постійні коефіцієнти, знак $\sum_{k-1}^{m} \oplus$ - означає суму всіх елементів за модулем. При відповідному виборі коефіцієнтів a_k генерується послідовність максимального періоду.

1.2. Аналіз існуючих рішень

Одним з методів формування ПЧ є рекурентний алгоритм формування послідовності ПЧ [4]. Псевдовипадковою послідовністю (ПП) вважається послідовність $\xi_1, \xi_2, \ldots, \xi_n$, що визначається за деяким рекурентним правилом:

$$\xi_n = f(\xi_{n-1}, \xi_{n-2}, \ldots, \xi_{n-k})$$

так, що при $k \le n \le N$ наділена статичними властивостями незалежно вибраних значень рівномірно розподіленої випадкової величини. В такому алгоритмі кожне наступне значення є результатом обчислення функції, яка залежить від певної кількості попередніх членів. Важливо відмітити, що на деякому кроці можливе отримання нулів у всіх розрядах. Тому потрібно забезпечувати відкидання таких ситуацій в процесі рішення задачі.

В деяких машинах застосовується інший алгоритм генерування ПП. Будуються дві послідовності цілих чисел відповідно рекурентному співвідношенню:

$$U_{n+1} \equiv 3U_n (mod\ p),$$

$$V_{n+1} \equiv 7V_n (mod\ p).$$

В якості p обирається значення $p=2^{31}-1$. Числа U_n та V_n знаходились в діапазоні $1 \leq A \leq p$ і були записані в двійковій системі числення. Після цього обиралось число W_n, що отримувалось за допомогою перестановки двійкових знаків числа V_n у зворотньому порядку. Над числами U_n та W_n виконується порівняння або порозрядно додаються по модулю два. Отримане значення приводять в діапазон від 0 до 1 та приймають в якості значення ξ_n члена ПП.

Лінійний конгруентний метод формування ПЧ базується на обчисленні лінійної рекурентної послідовності по модулю деякого натурального числа m, за формулою:

$$X_{k+1} = (aX_k + c) \bmod m,$$

де a, c, m - деякі числові коефіцієнти. Отримувана послідовність залежить від початкового числа і значень коефіцієнтів. Також є можливою зміна періоду генератора. При виборі певних значень коефіцієнтів, період стає досить великим, щоб задовольнити умови випадковості. Даний метод використовується по сьогоднішній день у популярних мовах програмування, таких як Java, C, C++ та інших.

Одним з відомих алгоритмів генерації ПЧ є метод Фібоначчі, який ще називають адитивним. Часто використовується в алгоритмах, критичних до якості випадкових чисел. Такі генератори природно реалізуються у арифметиці дійсних чисел, а швидкість виконання арифметичних операцій з числами є близькою до швидкості операцій з цілими числами. Даний метод формування заключається в тому, що наступний член послідовності залежить більше, ніж від одного попереднього значення. В класичному варіанті, крім

двох перших членів, кожен наступний член послідовності розраховується як сума двох попередніх. Одним з варіантів формули пошуку є:

$$X_n=(X_{n-24}+X_{n-55})mod(2^m),$$

де $n \geq 55$ – порядковий номер числа в послідовності, m – довільне парне число, а $X_0, X_1, \ldots, X_n$ – довільні цілі числа (з яких не всі парні). Перевагами алгоритму є швидкість, так як не потребує множення чисел, і досить великий період, але випадковість чисел в даній послідовності є малодослідженою.

Ще одним з поширених є генератор Фібоначчі на базі ітеративної формули

$$X_k = \begin{cases} X_{k-a} - X_{k-b}, & X_{k-a} \geq X_{k-b} \\ X_{k-a} - X_{k-b} + 1, & X_{k-a} < X_{k-b} \end{cases}$$

де X_k – дійсне число в інтервалі [0,1), a,b – цілі додатні числа. Для роботи потрібно знати max {a, b} попередніх сформованих ВЧ.

В програмних реалізаціях використовуються циклічні черги на базі масиву для зберігання вже згенерованих ВЧ. Зазвичай беруться значення (a, b) = (55, 24), (17, 5) або (97, 33). Чим більші константи, тим вища розмірність простору, в якому сформовані ВЧ матимуть рівномірний розподіл, а також тим більша потреба пам’яті для роботи алгоритму. Описаний алгоритм зі значеннями (a, b) = (20, 5) використовується в системі автоматизації математичних розрахунків MatLab.

Вихор Мерсена – ГПЧ на основі властивостей простих чисел Мерсена, який забезпечує швидку генерацію ПЧ і позбавлений таких недоліків як малий період, передбачуваність, легко виявляємі статистичні залежністі. Такий генератор не відповідає критеріям криптостійкості, тому не може використовуватись у криптографії [5].

Вихор Мерсена представляє собою циклічний регістр зсуву з узагальненим зворотнім зв’язком. Цей алгоритм забезпечує рівномірний розподіл ПЧ в 623 вимірах, а лінійні конгруентні методи – в 5. Тому кореляція двох послідовностей вибірок вихідної послідовності такого методу дуже мала.

Вихор Мерсена має великий період, що рівний числу Мерсена 219937-1. Цього достатньо для багатьох задач.

Основою метода є рекурентне співвідношення

$$X_{k+n} = X_{k+m} \oplus \left(X_k^u \middle| X_{k+1}^l\right)A, (k = 0,1, \dots),$$

де n – ціле число, яке визначає ступінь рекурентності, m – ціле число, при чому $1<m<n$, A – матриця розміром WxW. X_k^u, X_{k+1}^l позначають старші wr і молодші r біт. Даний метод реалізований у бібліотеці gLib, мовах PHP, Python і Ruby.

Генератори М-послідовностей (послідовностей максимального періоду) мають відносно просту реалізацію що програмну, що апаратну. Будуються на основі регістру зсуву з лінійним зворотнім зв'язком. Закон розподілу отриманої послідовності буде близьким до рівномірного. Для отримання послідовності максимального періоду необхідно і достатньо, щоб характеристичний многочлен, який описує схему зворотнього зв'язку, був примітивним за модулем 2. При цьому ж формування примітивних многочленів для характеристичного рівняння генератора є досить складним та вимагає значних обчислювальних ресурсів.

В генераторах ПЧ на основі регістрів зсуву з лінійним зворотнім зв'язком основною частиною був ланцюг зв'язків, що представляв собою суматор за модулем 2, де операндами виступали певні біти регістра зсуву. Така структура може знаходитись у 2^n станах. Нульовий стан буде виродженим, так як в наступних тактах не відбуватиметься зміни значень. Виходячи з цього, кількість можливих станів у регістра даної структури – 2^n-1. При певній конфігурації період роботи буде максимальним, що означає можливість отримання М-послідовності. Також очевидно, що структура послідовності та її період не залежать від початкового стану регістра. При цьому, так як елементи пам'яті мають парну кількість стійких станів, то величина періоду даного генератора буде непарною.

М-послідовності мають деякі властивості. Періодом послідовності максимальної довжини визначається як $T_M = r^n - 1$, де r – кількість стійких

станів елементів пам’яті, а n – довжина регістра. Тоді при трьох станах і довжині регістра 3 величина максимального періоду буде 26. Якщо послідовність містить в собі деяку фіксовану кількість елементів, то кількість одиничних серій визначеної ваги буде рівна кількості нульових серій цієї ж ваги.

Суму за модулем 2 елементів двох М-послідовностей, зсунутих між собою на довільну кількість бітів, але меншу за період, також представляє собою М-послідовність, але з зсувом на іншу кількість бітів. Якщо сумувати за модулем 2 М-послідовності різного періоду, то отримана послідовність також буде М-послідовністю, при цьому її період буде рівний найменшому спільному кратному величин періодів вхідних послідовностей.

Регістра зсуву являють собою ланцюжок тригерів, послідовно з’єднаних між собою. Основний режим їх роботи заключається в зсуві розрядів числа, що знаходиться в тригерах. В кожній ітерації значення попереднього розряду перезаписується до наступного по порядку і ланцюгу тригерів. Значення з кожним приходом тактового сигналу зміщуються або в ліву, або ж в праву сторону.

При моделюванні станів модулів системи доцільне використання генератора рівномірно розподілених рівновагових двійкових векторів. Основним недоліком існуючих рішень є повтор векторів у сформованій послідовності.

1.3. Обґрунтування теми дипломного проекту

ВБС є дуже поширеними в обчислювальних системах (паралельні та розподілені обчислення) і різних сферах управління складними комп’ютерними системами (космічними, авіаційними, морськими, атомними та іншими). Раніше основною сферою застосування таких систем була тільки наукова, так як в ній вимагаються потужні обчислювальні ресурси. Зараз багатопроцесорні обчислювальні системи (БОС) застосовуються також і в бізнесі, і веденні документообігу [6].

Ефективність функціонування різних БОС в значній мірі залежить від проведених заходів для забезпечення системної діагностики. Тобто від виявлення, усунення неполадок приладів, вчасне відключення чи реконфігурація системи з урахуванням напрямків інформаційних потоків даних між процесорами. Псевдовипадкове тестування є одним з перспективних напрямків в області технічної діагностики. Основним плюсом є спрощення всієї процедури тестування, яке пов'язане з найбільш трудомістким етапом – синтезом тесту для заданого класу несправностей. Ще одним фактором є успішний розвиток методів синтезу схеми, які орієнтовані на особливості такого методу тестування. Основними характеристиками даних систем є відношення ціни до обчислювальної потужності, надійність, можливість масштабування системи та сумісність з програмним забезпеченням (ПЗ). Відмова такої системи може призвести до катастрофічних наслідків, тому надійність роботи є дуже важливою характеристикою.

Надійність – це властивість об'єктів зберігати здатність виконувати потрібні функції протягом заданого періоду часу. Надійність є комплексною властивістю, яке в залежності від призначення об'єкту та умов його застосування може включати в себе відмовостійкість, довговічність, можливість ремонту або їх комбінації.

Відмовостійкість – це властивість об'єктів зберігати працездатність після відмови одного або декількох елементів. Відмовостійкість визначається кількістю будь-яких послідовних одиничних відмов компонентів, після яких зберігається працездатність системи в цілому. Базовий рівень відмовостійкості включає в себе захист від відмови будь-якого одного елемента системи, після якої зберігається працездатність всієї системи. Основним методом підвищення відмовостійкості – надлишковість. Але потрібно звернути увагу на факт, що для забезпечення відмовостійкості системи введення апаратної надлишковості у більшості випадків зменшує надійність всієї системи, так як кількість

компонентів стає більшою. Також знижується і швидкодія, адже для проведення діагностики вимагається більш часу.

Відмовостійкість може бути забезпечена активною чи пасивною схемами. За пасивною схемою забезпечення відмовостійкості заключається в тому, що система е втрачає свої функціональні можливості у випадку відмови певних елементів, тобто відмова «маскується» системою. Активна схема має на увазі виділення процесів виявлення відмови, її локалізації та відповідної реконфігурації системи.

В наш час дуже часто ВБС проектуються саме за активною схемою забезпечення відмовостійкості, так як ставляться підвищені вимоги до надійності системи. Деякі компоненти системи дублюються або мажоруються. Такі системи мають можливість самотестування, тому що стан всіх модулів системи має бути відомий у будь-який момент часу для вчасного виявлення відмов елементів і реконфігурації при конкретних умовах. При цьому інші ж частини системи мають виконувати всю функціональність несправних модулів.

Довговічність – властивість об’єктів зберігати працездатний стан до настання критичного стану при певних умовах експлуатації. До критеріїв даної властивості відносяться фізичний знос системи (коли подальший ремонт і експлуатація системи стають невигідними, так як витрати перевищують дохід в експлуатації) і моральний знос – невідповідність параметрів елемента або системи сучасним технологіям їх експлуатації. До показників довговічності відносяться середній ресурс (математичне сподівання ресурсу), гама-відсотковий ресурс (сумарне напрацювання, протягом якого об’єкт не досягне критичного стану з заданою ймовірністю), призначений ресурс (сумарне напрацювання, при досягненні якого експлуатація об’єкту має бути припинена незалежно від його технічного стану), середній термін роботи (математичне сподівання терміну роботи), гама-відсотковий термін роботи (календарна тривалість експлуатації об’єкта, протягом якої він не досягне критичного

стану з заданою ймовірністю) і призначений термін використання(календарна тривалість експлуатації, при досягнені якої використання має бути припинене).

Розрахунок надійності є важливим на початкових етапах розробки, так як це застерігає від додаткових витрат коштів при виявленні помилок на наступних стадіях. Можливим шляхом рішення даної проблеми є проведення статистичних експериментів з моделями. Початковими даними для розрахунку є ймовірність безвідмовної роботи модулів системи протягом заданого проміжку часу та модель поведінки системи, яка демонструє залежність стану ВБС від стану її елементів [7].

Надійністю програмного забезпечення називають властивість програми, яка виражається у виконанні вказаних функцій в певних умовах роботи і на заданій обчислювальній системі. Тобто, дане поняття визначається аналогічно до того, яку суть має таке поняття як надійність апаратури. Але механізми виникнення відмови ПЗ і відмови апапратної частини суттєво відрізняються один від одного. Якщо причиною відмови апаратури є, як правило, руйнування якихось її елементів, то відмова програми – невідповідністю реалізації даного ПЗ поставленій задачі. Ця невідповідність виникає з двох причин: або було допущено порушення технічних вимог до ПЗ самими розробниками, або ж специфікація була сформовано неточно або недостатньо повною.

Для дослідження відмовостійкості можуть використовуватись графо-логічні моделі ВБС (GL-моделі) [8]. Вони дозволяють проводити аналізувати поведінку ВБС, які містять велику кількість процесорів, доступними сучасними засобами. Ці моделі поєднують між собою властивості булевих функцій і графів. Для роботи з ними корисною є згенерована програмно або апаратно послідовність псевдовипадкових векторів стану системи, в яких за допомогою булевих змінних задається працездатність кожного модуля (значення 0 – система у робочому стані, 1 – система вийшла з ладу). В залежності від рівня деталізації побудови моделі поведінки елементами ВБС

можуть вважатись окремі підсистеми, процесори, мікросхеми та інше. Проводять аналіз вихідної послідовності, реакції системи з метою виявлення неполадок або аварійних завершень роботи системи. За допомогою визначення ймовірностей появи векторів, при яких система залишається працездатною, можна розрахувати показники надійності ВБС.

2. АНАЛІЗ СПОСОБІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

2.1. Способи формування послідовності двійкових векторів

Одним з відомих способів формування послідовності РДВ є використання регістру зсуву з лінійним зворотнім зв’язком. У такого генератора значення вхідного біту дорівнює значенню лінійної БФ, яка залежить від поточного стану інших бітів регістру до зсуву. Може бути реалізований як програмно, так і апаратно. Такий генератор поділяється на 2 частини: сам регістр та схема зворотнього зв’язку [9].

Основні положення теорії та принципи щодо можливостей формування послідовностей ПЧ за допомогою структур на основі регістру зі зворотнім зв’язком були розглянуті вперше в середині шістдесятих років в роботах Селмера, Голомба, Гілла. Після цього дослідження у вказаному напрямку проводились такими авторами, як Федоров, Яковлев, Добрис, Бухараєв, Ярмолик та інші.

Регістр складається із функціональних комірок пам’яті, кожна з яких зберігає поточний стан одного біта [10]. Кількість комірок є довжиною регістра. Комірки, зазвичай, нумеруються числами 0, 1, 2 і т.д. Значення нового біта після зсуву визначається ще до зсуву та записується в нульовий розряд. Загальна структура такого регістру зсуву зображена на рис. 3.

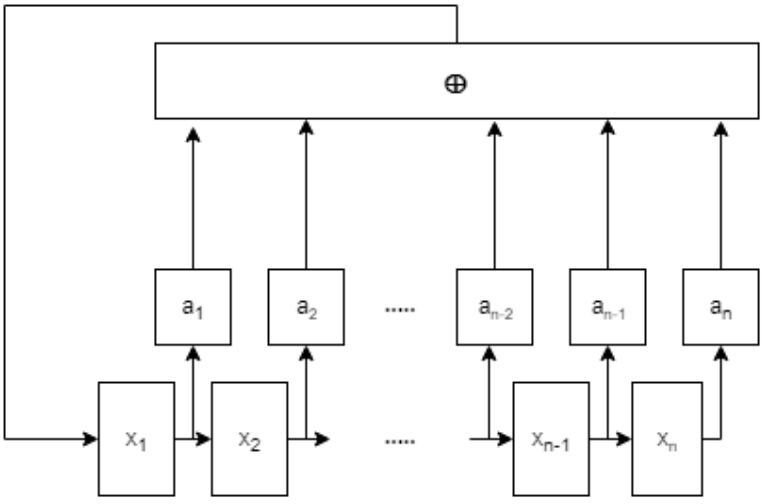

Рисунок 3 – Регістр зсуву з лінійним зворотнім зв’язком

В даному випадку зворотнім зв'язком виступає суматор за модулем 2. Операндами суматора виступають визначені біти регістру зсуву. Елементи пам'яті X_i, $i=1,2,\ldots,n$ утворюють регістр зсуву, а стан блоків a_i задає конкретну конфігурацію ланцюжка зворотнього зв'язку, яка в свою чергу визначається числом розрядів. Така структура може знаходитись у 2^n станах, при чому виродженим випадком буде нульових стан, так як є незмінним за будь-якої обраної конфігурації. Тому даний формувач може видати послідовність, максимальний період якої буде рівний 2^n-1. Вихідні послідовності таких генераторів й отримали назву М-послідовностей [11].

Елементи $a_i \in \{0,1\}$, $i = 0,1,\ldots,n$ визначають конкретну топологію схеми лінійного зворотнього зв'язку, можуть бути описані за допомогою матриці наступного вигляду:

$$M_a = \begin{Vmatrix} a_1 & a_2 & \ldots\ldots\ldots & a_{n-2} & a_{n-1} & a_n \\ 1 & 0 & \ldots\ldots\ldots & 0 & 0 & 0 \\ 0 & 1 & \ldots\ldots\ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots\ldots\ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots\ldots\ldots & 1 & 0 & 0 \\ 0 & 0 & \ldots\ldots\ldots & 0 & 1 & 0 \end{Vmatrix}$$

При цьому характеристична матриця M_a^x розраховується на основі матричного рівняння вигляду:

$$M_a^x = M_a \oplus x \cdot E,$$

Де E – одинична матриця порядку n. Тобто:

$$M_a^x = \begin{Vmatrix} a_1 + x & a_2 & \ldots\ldots\ldots & a_{n-2} & a_{n-1} & a_n \\ 1 & x & \ldots\ldots\ldots & 0 & 0 & 0 \\ 0 & 1 & \ldots\ldots\ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots\ldots\ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots\ldots\ldots & 1 & x & 0 \\ 0 & 0 & \ldots\ldots\ldots & 0 & 1 & x \end{Vmatrix}$$

Циклічні властивості такого регістру зсуву визначаються характеристичним рівнянням, як отримується під час знаходження визначників матриці. Таким чином при розкладі матриці по елементам першого рядка буде отримано:

$$\phi(x) = 1 + \sum_{i=1}^{n} a_i \cdot x^i$$

Для отримання послідовності з максимальним періодом необхідно і достатньо умови, що характеристичний многочлен, що описує схему зворотнього зв'язку, був примітивним за модулем 2. Степінь даного многочлена визначатиметься кількістю розрядів регістра. У такому випадку многочлен *n*-го степеня буде примітивним, якщо він одночасно є дільником величини $x^{2^{n-1}} + 1$ і не є дільником $x^r + 1$ для всіх *r*, які є дільниками $2^n - 1$. Для знаходження примітивних многочленів заданого степеня потрібні значні обчислювальні ресурси. Задача є подібною до задачі визначення чи є достатньо велике число простим.

В таких ПП частота появи групи однакових значень бітів має зменшуватись зі збільшенням числа елементів послідовності. Серед всіх серій однакових елементів половина відповідає групам одиниць, а інша половина – нулів. Якщо послідовність містить деяке фіксоване число елементів, то кількість серій одиниць та кількість нульових серій будуть однаковими.

М-послідовності наділені всіма властивостями послідовностей чисто випадкового генератора, при чому зі збільшенням довжини періоду статичні характеристики стають все більш близькими до аналогічних характеристик випадкових послідовностей. Особливістю М-послідовностей є можливість їх відтворення.

Якщо просумувати декілька М-послідовностей з різним періодом, то в результаті буде отримана послідовність з періодом, що дорівнює найменшому спільному кратному всіх величин періодів вхідних послідовностей.

Для синтезу схеми ГПЧ може бути використана вибірка по певному правилу елементів М-послідовності з метою зменшення кореляції між бітами вихідної послідовності багаторозрядних наборів. Реалізація такої виборки відбувається за рахунок структурної надлишковості схеми у вигляді додаткових суматорів за модулем 2.

Функцією зворотнього зв'язку таких генераторів є лінійна БФ, яка залежить від декількох або всіх розрядів регістра. Найчастіше знаходять використання такі логічні операції, як кон'юнкція, диз'юнкція та сума за модулем два.

Описаний генератор є досить швидким і дозволяє отримати послідовність ПЧ максимального періоду, яка включає в себе вектори різної ваги.

Ще один варіант, запропонований Льюїсом і Пейном у 1973 році, розглядає регістр зсуву з узагальненим зворотнім зв'язком. Є покращенням генератора з лінійним зворотнім зв'язком. Ідея заключається в тому, що рахується декілька значень лінійної функції та записується не в один біт, а відповідно зразу в декілька обраних бітів. Перевагами даного методу є:

- швидкість роботи;
- можливість отримання слова різної довжини;
- багатовимірний рівномірний розподіл.

Недоліками є:

- для ініціалізації генератора з великою кількістю розрядів потрібно багато часу;
- за період генератора деякі значення послідовності будуть повторюватись.

Структурний метод формування послідовностей РДВ будується на основі регістру зсуву зі зворотнім зв'язком. Для виключення можливості повторів двійкових векторів пропонується використання двох ГПЧ в структурі формувача. Перший з них формує однобітну послідовність великого періоду,

який повинен бути значно більшим тривалості робочого циклу. А другий ГПЧ є регістром зсуву з управляючим лінійним зв’язком. В залежності від стану генератора він може працювати в двох режимах. Такий метод значно підвищує продуктивність системи та знижує складність схемотехнічної реалізації [12].

2.2. Порівняння мов програмування для реалізації

В наш час існує багато мов програмування високого і низького рівнів різного призначення, за допомогою яких можна реалізувати потрібний додаток. До них відносяться:

- мова Асемблера;
- C;
- C#;
- Java;
- Python;
- C++ та нші.

Мова Асемблера – машинно-орієнтована мова програмування низького рівня. Команди даної мови мають пряму відповідність до машинних команд або їх послідовності. Є платформозалежною, тобто специфічною для конкретної архітектури, операційної системи та варіанту синтаксису мови. Мова Асемблера використовується для написання програм або їх частин у випадках, коли важливі швидкодія та об’єм використовуваної пам’яті. Результатом багатьох компіляторів з мов високого рівня є програма на мові Асемблера.

Мова знаходить застосування при написанні драйверів, ігор, мікроконтролерів, систем із обмеженим ресурсом пам’яті, вірусів, антивірусів та іншого. Програми, написані мовою Асемблера, мають високу швидкодію та потребують малі витрати ресурсів пам’яті. При цьому ж мають погану читабельність коду, є складними для підтримки, реалізації будь-яких складних дій та несумісними з іншими платформами.

Мова С створювалась з метою полегшення написання великих програм на зменшення кількості помилок, що допускаються при написанні коду. Є першою мовою високого рівня, яка змогла конкурувати з мовою Асемблера при розробці системного ПЗ. Також мова С має високий поріг входження, не призначений для веб-розробки, не є надійним. Є потужним інструментом для створення складних програм і в той же час надає можливість створення зовсім ненадійних програм з непередбачуваною поведінкою [13].

C# – об'єктно-орієнтована мова програмування високого рівня. Розроблена в 1998-2001 роках групою інженерів-програмістів під керівництвом Андерса Хейлсберга компанією Майкрософт як мова розробки додатків для платформи Microsoft .NET Framework. Надалі була стандартизована як ECMA-334 та ISO/IEC 23270. Відноситься до групи мов з C-подібним синтаксисом, найбільш схожим з C++ і Java.

Має статичну типізацію даних, підтримує поліморфізм, перевантаження операторів (оператори явного та неявного перетворень також), делегати, атрибути, події, узагальнені типи, замикання, виключення та інше. Забравши багато переваг від популярних в той момент мов, C# виключає деякі проблематичні при розробці ПЗ моделі. Наприклад, не підтримується множинне наслідування класів. Розроблялася як мова програмування прикладного рівня для CLR, тому є сильно залежною від можливостей самого CLR. Недоліками даної мови є:

- орієнтованість в основному на .NET (Windows-платформу);
- відсутність безкоштовної ліцензії на використання для компаній;
- при обробці великої кількості даних значно програватиме по швидкості мові C++.

Java – строго типізована об'єктно-орієнтована високорівнева мова програмування загального призначення. Мова є надійною, незалежною від платформи, підтримує паралельні обчислення. При цьому недоліками є:

- платне комерційне використання;

- підвищені вимоги до об’єму оперативної пам’яті;
- відносно низька продуктивність;
- багатошаровий складний код.

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника ПЗ, читабельність коду та розробку веб-додатків. Є лідером в області аналізу великих об’ємів даних, застосовується в науковій сфері та машинному навчанні. Його філософія дизайну демонструє читабельність коду, а синтаксис дозволяє розробникам ПЗ вміщувати реалізацію алгоритмів у меншій кількості коду, ніж іншими популярними в даний момент мовами програмування. Підтримує декілька парадигм програмування. В тому числі, об’єктно-орієнтовану, імперативну, функціональну та процедурну.

Має динамічну типізацію даних і управління пам’яттю здійснюється автоматично. Має велику стандартну бібліотеку. Є досить повільною мовою навіть при порівнянні з іншими інтерпретованими мовами. Для організації швидких кодів потрібно створювати максимально логічний та простий код, що вимагає наявність практики та нестандартного мислення.

C++ – статично типізована мова програмування загального призначення. Підтримує такі парадигми програмування як процедурна, узагальнена. Найбільше уваги виділено підтримці об’єктно-орієнтованого програмування. Забезпечує модульність, окрему компіляцію, обробку помилок, абстракцію даних, оголошення типів, віртуальні функції. Стандартна бібліотека є досить обширною, в тому ж числі, включає в себе загальновживані контейнери й алгоритми. Дана мова успішно поєднує в собі особливості як мов низького рівня, так і високого.

Є однією з найпопулярніших мов. Використовується дана мова у сферах створення операційних систем, різних прикладних програм, драйверів, серверів, ігор. Плюсами даної мови є:

- висока сумісність з мовою C;

- відносна простота та надійність управління пам’яттю;
- висока обчислювальна потужність;
- підтримка різних стилів програмування;
- можливість вбудови предметно-орієнтованих мов програмування в основний код.

Qt –фреймворк для розробки ПЗ, в основному, мовою програмування C++ [14]. Є повністю об’єктно-орієнтованим. Підтримає багато різних платформ. Також є можливість використання з мовами Python (PyQt), Ruby (QtRuby), Java (Qt Jambi) та іншими. Дана бібліотека дозволяє запускати створені з її використанням додатки на більшості операційних систем шляхом простої компіляції програми без змін у коді. Включає в себе функціональність для розробки прикладного програмного забезпечення, починаючи з графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних, XML. Ще одною перевагою даної бібліотеки є добре продуманий, логічний набір класів, що надає високий рівень абстракції при написанні коду. Тому код стає більш читабельним і швидкість розробки програмного продукту зростає.

2.3. Мова програмування C++. Бібліотека Qt

C++ створювалась як розширення до мови С Б’ярне Страуструпом на початку вісімдесятих років минулого століття. В той момент мова С була основною мовою UNIX-систем і є швидкою, багатофункціональною і переносною. Страуструп додав до неї можливість роботи з класами й об’єктами, що й породило розвиток нової мови C++.

Б’ярне планував реалізувати розширення до мови С, але з часом функціональність значно розширилась і стала окремою мовою програмування, яка має можливості низькорівневого доступу до пам’яті, ООП, універсального й функціонального стилів, підтримки класів. Розроблялась C++ з основним напрямком для системного програмування, вбудованого ПЗ з обмеженими ресурсами та великих систем, де важливі продуктивність, ефективність та

гнучкість використання. Наприклад, сервери, прикладні додатки, комутатори, космічні зонди, комерційні застосунки та інше.

C++ наслідує більшість синтаксису від попередника. Так само підтримує чотири типи управління пам’яттю, такі як зберігання статичних об’єктів, динамічних об’єктів, потоків та автоматичне зберігання об’єктів. Спосіб зберігання визначає тривалість життя об’єкта. При цьому якщо об’єкт не існує, але залишились вказівники на нього, то його значення вважається невизначеним. Підтримуються такі типи даних, як символьний, цілі числа знакові та беззнакові, з плаваючою крапкою та логічний.

Під час наслідування класу від інших, то наслідується реалізація класу, а також можна додавати нову функціональність у дочірньому класі або ж перевизначати існуючу. Підтримується множинне наслідування. Під час роботи програми спочатку спрацьовує конструктор базового класу, після цього конструктори нестатичних типів даних. Деструктор працюватиме в оберненому порядку. Наслідування може бути видимим повністю, захищеним, а також повністю закритим.

Метою поліморфізму є створення одного класу для задання назв всіх спільних дій декількох класів. Виконання кожної конкретної дії буде залежати від визначеного типу даних, на базі об’єкту якого викликається метод. Перевагою даної властивості ОО є те, що це знижує складність програм дозволяючи використовувати один і той же інтерфейс для однакових дій.

Інкапсуляція є в мові C++ реалізується через рівні доступу до елементів класу. Можливі такі як відкриті, захищені та закриті.

Шаблони програмування дозволяють застосовувати підхід загального програмування. Можлива параметризація фіксованим часом компіляції, типами або іншими шаблонами. Цей інструмент дозволяє оптимізувати код, але вимагає додаткових затрат, таких як величина коду, пам’ять.

Також підтримується використання класів, що дозволяє використовувати об’єктно-орієнтований підхід програмування. Класи включають в себе інкапсуляцію, поліморфізм, абстракцію та наслідування.

Стандартна бібліотека мови включає в себе різні структури даних такі, як списки, множини, черги, кортежі, алгоритми (пошуку, сортування), засоби вводу й виводу інформації, засоби для роботи з файлами, вказівники, регулярні вирази, багатопоточність, атомарність. Частина стандартної бібліотеки бере за основу стандартну бібліотеку шаблонів, яка включає в себе контейнери, ітератори, різні алгоритми.

Бібліотека Qt починала існування як платформонезалежний засіб для створення графічних інтерфейсів програм, написаних мовою C++. Зараз переросла в цілісний фреймворк, що дозволяє в більшості випадків при написанні програми використовувати тільки рідні класи Qt, а також бути практично повністю незалежним від платформи реалізації. До бібліотеки входять такі модулі як:

- класи ядра бібліотеки (QtCore);
- компоненти графічного інтерфейсу (QtGui);
- клас для класичних програми на основі віджетів (QtWidgets);
- для підтримки QML (QtQML);
- набір класів для мережного програмування (QtNetwork);
- для роботи з базами даних (QtSql);
- довідкова система (QtAssistant);
- класи підтримки модульного тестування (QtTest);
- роботи з аудіо- та відеофайлами (Phonon);
- підтримки повнотекстового пошуку (QtCLucene) та багато інших.

Qt є повністю інтегрованим середовищем розробки. В ньому надаються для проектування потрібні інструменти, а також для розробки досить складних систем, що використовуватимуться в прикладних додатках або мобільних платформах.

Одним з переваг є надання можливості розробникам працювати над проектом на різних операційних системах, використовуючи спільні інструменти та середовище налагодження. Інформація, яка потрібна компілятору, вказується в налаштуваннях збірки та запуску проекту. QtCreator дозволяє створити новий проект або імпортувати вже існуючий. Також формує всі необхідні файли в залежності від типу проекту, що створюється.

QtCreator вміє збирати та запускати код, що відрізняє його від звичайного текстового редактора. Він розуміє такі мови, як C++ та QML. Це надає можливість писати гарно форматований код, передбачувати що планується писати та надавати підказки, доповнювати написане автоматично, відображати повідомлення про помилки, попередження, локально перейменовувати частини тексту, демонструвати місце оголошення чи реалізації методів, класів.

Також надає підтримку збірки й запуску проекту на різних ОС. Налаштування збірки дозволяють легко переключатись між обраними системами. Якщо мобільний пристрій підключений до комп’ютера, то QtCreator формує пакет інсталяції, проводить інсталяцію на зовнішній пристрій і запускає програму.

Ще підтримує декілька систем контролю версій: Git, Subversion, Perforce, CVS і Mercurial. Доступна функціональній залежить від системи управління, але базові функції доступні всім. Підтримується порівняння файлів з останньою версією, відображення різниці, перегляд історії змін, анотацій файлів, прийнята та відміна змін.

В Qt доступні такі зручні для даної програмної реалізації структури даних, як QString, QBitArray, QSet, QMainWindow.

Vector – це клас, доступний у мові C++, який являє собою послідовний контейнер, який інкапсулює всередині масиви змінної довжини. Елементи структури зберігаються неперервно, що дає можливість доступу не тільки через ітератори, а й зміщення, які додаються до вказівника на вектор. Роботу

з пам’яттю виконується автоматично, розширюючи чи звужуючи виділену область при необхідності.

QString – це клас, що описаний в бібліотеці Qt. Призначений для зберігання і роботи з рядками з шістнадцяти-бітних символів Unicode. Даний клас неявно застосовує спільне використання даних (копіювання при записі) для запобігання лишньому виділенні пам’яті та непотрібного копіювання інформації. Це також запобігає лишнім витратам пам’яті у порівнянні з восьми-бітними словами. Даний клас застосовується в Qt API і Unicode забезпечує легкий переклад програми при потребі. Надає множину перевантажених функцій для спрощення роботи з рядками.

QBitArray – клас управляє бітовим (булевим) масивом. Кожне збережене значення займає лише один біт, що не витрачає лишню пам’ять. Використовується для збереження великої кількості змінних типу bool. Для операцій з бітами клас надає методи для запису setBit(), для зчитування testBit(), також оператор [], за допомогою якого можна звертатись до кожного біту окремо.

QSet – це контейнер, який записує елементи в деякому порядку і надає можливість дуже швидкого перегляду значень та виконання з ними операцій, характерних для множин, такі як кон’юнкція, диз’юнкція, різниця. Необхідною умовою є те, що ключі елементів мають бути різними. Даний контейнер можна використовувати у якості невпорядкованого списку для швидкого пошуку даних.

QMainWindow – клас, який реалізує головне вікно програми, яке містить в собі віджети (меню, панелі інструментів, робочу область, рядок стану). Зовнішній вигляд уже підготовлений і його віджети розташовані по місцям, тому не вимагають додаткових дій з розміщенням. Даний клас полегшує створення графічного інтерфейсу програми і дозволяє швидко реалізувати потрібний вигляд програми.

Програма, яка написана з використанням даної бібліотеки є швидкою, читабельною, надійною, стабільною. Тому була обрана для реалізації ПЗ для пошуку функцій управління та формування послідовності РДВ.

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ФОРМУВАННЯ ПОСЛІДОВНОСТЕЙ РІВНОВАГОВИХ ДВІЙКОВИХ ВЕКТОРІВ

3.1. Структура генератора

В даній роботі пропонується генератор псевдовипадкових РДВ, який побудований на базі регістру зсуву зі зворотнім зв’язком. Принцип роботи даного регістру заключається в організації зсуву двійкового вектора вправо з урахуванням значення сигналу управління. Якщо значення сигналу управління дорівнює нулю, то всі розряди беруть участь у циклічному зсуві (значення першого розряд стає другим, другого – третім, …, останнього – першим). Якщо ж дорівнює одиниці – перший розряд не бере участі у зсуві (значення першого розряду залишається незмінним, другого стає третім, …, останнього – стає значенням другого розряду) [15].

Кількість розрядів регістру дорівнює n, що відповідає довжині векторів послідовності, p – вага кожного з векторів, $x_1, x_2,\ldots,x_n$ – булеві змінні, які відображають поточний стан відповідних розрядів регістру. В кожному такті роботи генерується двійковий вектор $X = (x_1, x_2,\ldots,x_n)$. Керуючі сигнали формуються на базі спеціально вибраних булевих функцій (БФ) затримки, які й визначають порядок векторів у згенерованій послідовності. Значення функції затримки f залежить від значень розрядів поточного вектора [16]. На рис. 4 зображена загальна структура генератора.

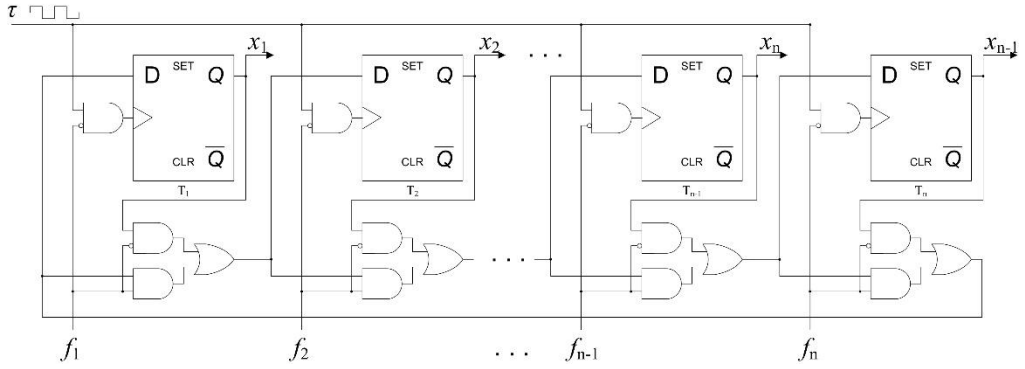

Рисунок 4 – Загальна структура генератора

Розглядається окремий випадок генератора, в якого функція затримки визначається тільки для першого розряду регістра [17]. І приймає значення одиниці, якщо перший розряд не бере участі в зсуві (тобто необхідно затримати) та 0, якщо циклічний зсув відбувається у всіх розрядах.

За допомогою даного генератора з будь-якого вектора заданих довжини та ваги можна отримати будь-який інший вектор тих самих ваги й довжини шляхом виконання операцій зсуву в право без чи з затримкою першого розряду.

Якщо у векторі значення першого й останнього бітів є однаковими, то результати операцій зсуву з затримкою та зсуву без затримки першого розряду будуть однаковими. Якщо ж значення цих бітів будуть різними, тобто $x_1 \oplus x_n = 1$, то результатом виконання зсувів з чи без затримки будуть різні вектори [18].

Тому при пошуку функцій управління для векторів, значення перших й останніх бітів яких однакові, значення можна залишати невизначеними, так як вони не вплинуть на результат.

Для двох векторів, у яких значення крайніх бітів поміняні місцями, значення функції затримки будуть однаковими. Це обумовлено надходженням векторів до однієї групи зсуву з затримкою, для запобігання зациклення.

Ще однією з властивостей даного генератора є те, що для будь-яких довжини й ваги множина функцій управління генератором, що дозволять сформувати М-послідовності, не є пустою.

Основною задачею при побудові даного генератора РДВ є пошук таких функцій, за допомогою яких можна отримати почергово всі вектори однакової ваги без повторень виключно за допомогою циклічного зсуву всіх розрядів вправо або всіх, крім першого.

Властивістю описаного генератора є можливість з довільного вектора довжини n ваги p отримати будь-який інший вектор тих же довжини й ваги за допомогою виконання зсувів вправо з та без затримки першого розряду. Це

надає підтвердження, що на основі запропонованого генератора можна отримати послідовно всі рівновагові двійкові вектори [19].

Також при роботі генератора для векторів, що мають однакові значення першого та останнього бітів, значення функції затримки є несуттєвим, тому що результати зсуву з затримкою та зсуву без затримки будуть однаковими. Це дає можливість значення функції управління для наборів з однаковими першим та останнім бітами залишати невизначеним.

3.2. Архітектура програмного забезпечення

Під час написання програми були описані 4 класи: Converter, FunctionGenerator, Register і . Відношення між вказаними класами можна побачити на рис. 5. Деякі класи зберігають в собі вказівники на об’єкти класів іншого типу.

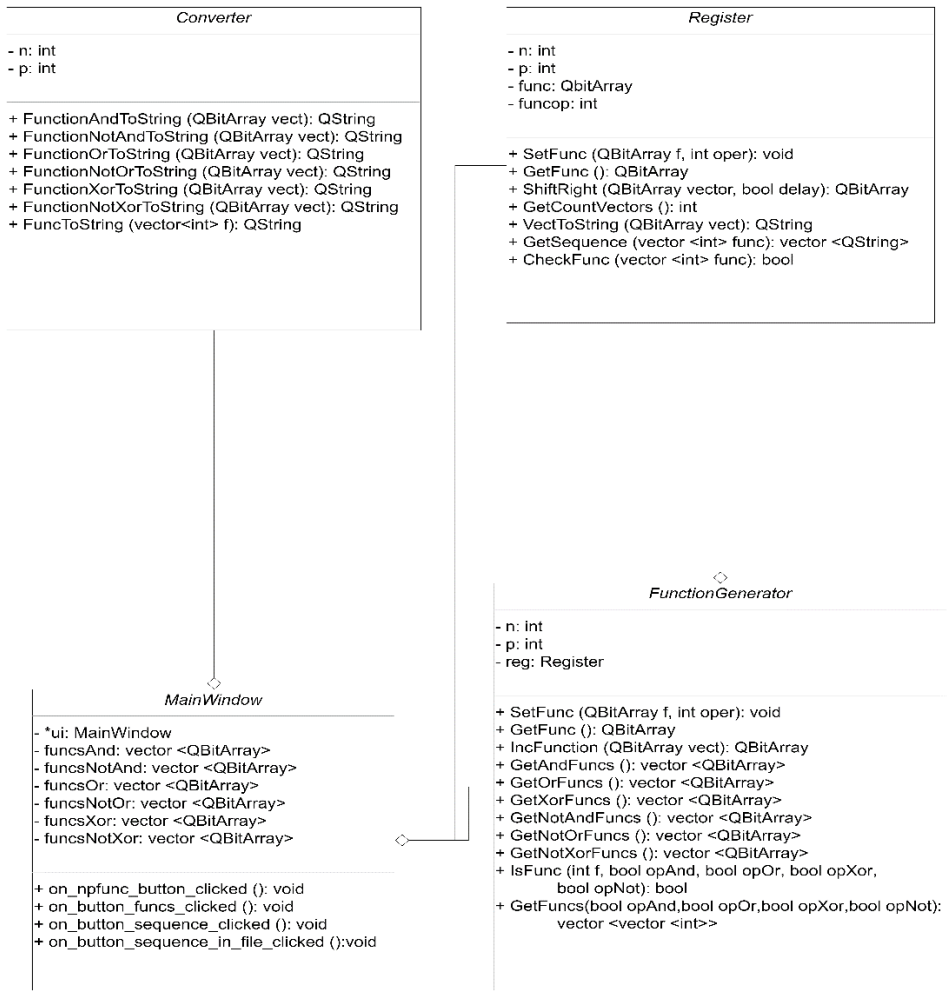

Рисунок 5 – Схема взаємодії класів.

При реалізації програми класи мали структуру:

- Converter – об’єкти даного типу дозволяють представити сформовану функцію у вигляді рядка. Методи даного класу дають можливість перетворення функцій управління як з використанням окремих логічних операцій, так і з допустимими декількома логічними операціями в одній функції, сформованих у вигляді масивів бітів (використовується структура даних QBitArray), у текстові рядки (використовується QString) для зручного виводу на робочий екран програми.
  - QString FunctionAndToString(QBitArray vect) – метод для формуваня рядка з функції, що вміщує в собі тільки логічну операцію І;
  - QString FunctioOrToString(QBitArray vect) – функція повертає рядок, що формується з отриманої функції, де допустима логічна операція АБО;
  - QString FunctionXorToString(QBitArray vect) – метод призначений для генерації рядка на основі переданої функції з операцією ВИКЛЮЧНЕ АБО;
  - QString FunctionNotAndToString(QBitArray vect) – метод повертає значення рядку, що вміщує функцію з операціями І-НЕ між операндами;
  - QString FunctionNotOrToString(QBitArray vect) – метод для формування функції з операціями АБО-НЕ на основі переданої функції;
  - QString FunctionNotXorString(QBitArray vect) – метод, що призначений для конвертації функції на базі операції ВИКЛЮЧНЕ АБО-НЕ у рядок;
  - QString FuncToString(vector<int> f) – даний метод формує рядок з вхідної функції, яка в залежності від значень елементів масиву може

вміщати в собі такі логічні операції як І, АБО, ВИКЛЮЧНЕ АБО та їх заперечення.

- FunctionGenerator – клас, який описаний для генерації функцій. Методи даного класу дозволяють задати поточну функцію управління регістром зсуву з прямим зворотнім зв’язком, отримати значення поточної функції управління, а також знаходити можливі ФУ з використанням тільки однієї логічної операції та з використанням декількох. Також є можливість перевірки булевої функції, представленої у вигляді масиву бітів, чи може вона бути ФУ для регістра заданих характеристик.
  - void SetFunc(QBitArray f, int oper) – метод задання функції управління регістром зсуву, за допомогою якої формуватиметься послідовність РДВ;
  - QBitArray GetFunc() – метод повертає функцію, яка в даний момент управляє першим розрядом регістру при зсуві;
  - QBitArray IncFunction(QBitArray vect); - даний метод формує наступну функцію управління для перевірки роботи регістра;
  - vector<QBitArray> GetAndFuncs() – даний метод знаходить всі ФУ, які включають в себе тільки логічну операцію І;
  - vector<QBitArray> GetOrFuncs() – цей метод перебирає всі можливі функції з операцією АБО та знаходить які можуть використовуватись як ФУ;
  - vector<QBitArray> GetXorFuncs() – метод для перевірки та пошуку ФУ регістром, де допускається тільки операція ВИКЛЮЧНЕ АБО;
  - vector<QBitArray> GetNotAndFuncs() – написаний метод знаходить ФУ, які містять в собі тільки операції І-НЕ;
  - vector<QBitArray> GetNotOrFuncs() – є аналогічним методом до попередніх, але в функціях операцією є АБО-НЕ;
  - vector<QBitArray> GetNotXorFuncs() – метод, що шукає функції з використанням заперечення функції ВИКЛЮЧНЕ АБО;

- bool IsFunc(int f, bool opAnd, bool opOr, bool opXor, bool opNot) – даний метод реалізований для перевірки функції з заданими параметрами чи є вона ФУ регістра, що дозволяє сформувати послідовність РДВ максимального періоду й без повторень;
- vector<vector<int>> GetFuncs(bool opAnd, bool opOr, bool opXor, bool opNot) – метод, що перебирає всі можливі функції на базі допустимих логічних операцій і повертає масив ФУ регістром, які задовольняють всім вимогам.

- Register – модуль програми, який моделює роботу регістра. Задається довжина й вага векторів, а також функція управління. Модуль демонструє результат роботи регістру заданої конфігурації. Методи дозволяють управляти характеристиками регістра, наприклад, встановлювати довжину, вагу, поточну функцію управління. Також в даному класі описані функції, що виконують зсув вправо з затримкою першого розряду або без затримки, що перевіряють булеву функцію з даних умов, що формують рядок для відображення поточного стану регістра для виводу користувачу.

  - void SetFunc(QBitArray f, int oper) – метод встановлює отриману функцію як ФУ першим розрядом регістра при зсуві вправо;
  - QBitArray GetFunc() – повертає функцію, яка управляє регістром у даний момент;
  - QBitArray ShiftRight(QBitArray vector, bool delay) – метод, результатом роботи кого є масив значень бітів регістра після зсуву з урахуванням значеня ФУ первого розряду;
  - int GetCountVectors() – обчислює кількість можливих векторів заданих довжини та ваги;
  - QString VectToString(QBitArray vect) – формує текстовий рядок зі значень розрядів регістра;

- vector<QString> GetSequence(vector <int> func) – повертає послідовність РДВ, яку буде сформовано за допомогою отриманої ФУ;
- bool CheckFunc(vector<int> func) – встановлює отриману функцію як ФУ і перевіряє чи сформована генератором послідовність буде мати максимальний період та вектори не повторюватимуться в робочому циклі;

3.3. Пошук функцій управління генератором

Спочатку для задання параметрів пошуку потрібно вказати обрані довжину регістра *n* і вагу векторів *p* у текстових полях N і P, що знаходяться зверху зліва в програмі. Вигляд робочого вікна програми видно на рис.6. Після цього користувач має обрати логічні операції, які зможуть використовуватись у БФ. Обраною користувачем має бути хоча б одна операція. У протилежному випадку, у полі виведення функцій буде видане повідомлення про потребу обрання. За це відповідають чек-бокси, де галочка означає, що операція є дозволеною. Наявні три бінарні операції, такі як І, АБО та ВИКЛЮЧНЕ АБО, а також унарна операція НІ.

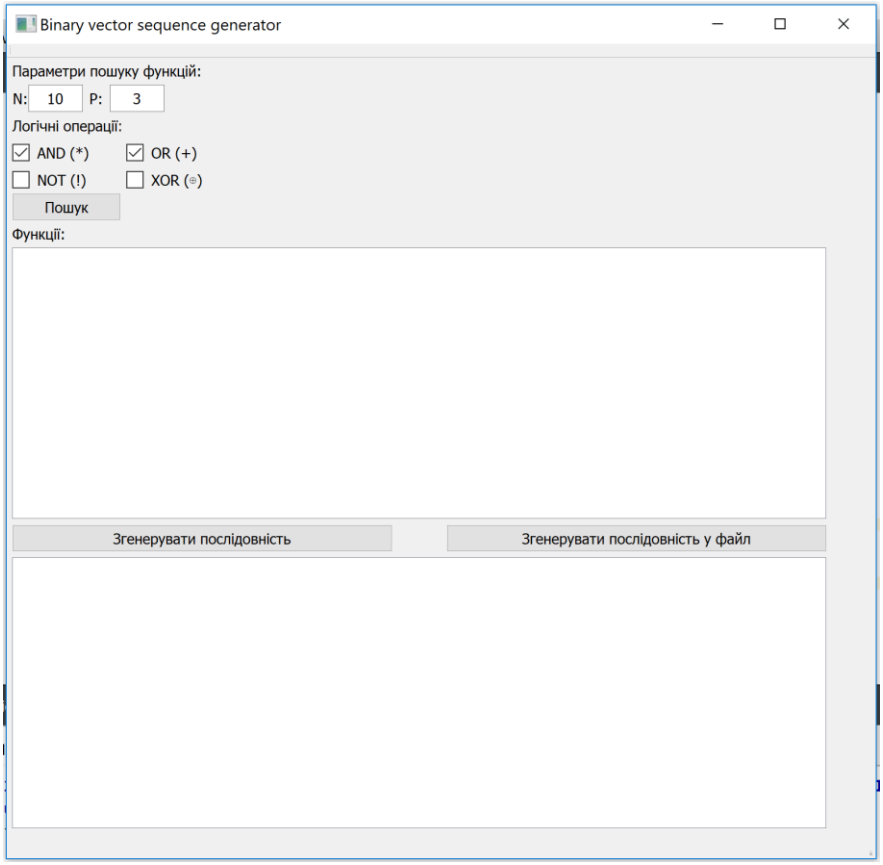

Рисунок 6 – Робоче вікно програми

Під час пошуку функцій можуть мати місце як функції тільки з одним видом логічних операцій, так і з декількома обраними. Всі знайдені БФ, які є коректними ФУ регістром зсуву для нашої задачі, будуть виведені у полі списку. У цьому списку функцій можна за допомогою кліку обрати функцію, після чого натиснувши кнопки «Згенерувати послідовність» або «Згенерувати послідовність у файл» сформувати послідовність обраною ФУ та за потребою записати її у файл в пам’яті комп’ютера.

Для пошуку всіх функцій був написаний алгоритм, який перебирає, в залежності від обраних параметрів пошуку, всі можливі функції, операндами якої є значення розрядів регістра, а логічні операції визначені користувачем. На рис. 7 видно, що починаємо ми зі значення 0, що задає таку ФУ, де для будь-якого вектора в поточний момент зсув виконуватиметься без затримки. Але даний варіант можливий лише у випадку ваги, рівній 1.

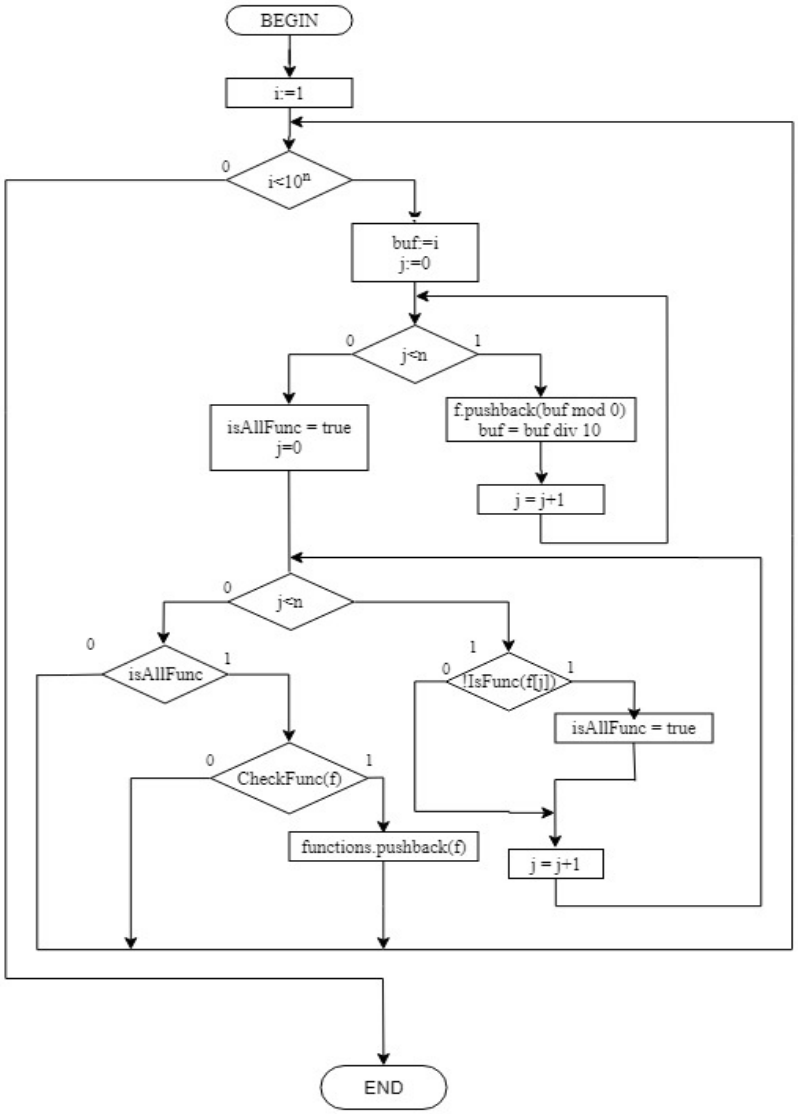

Рисунок 7 – Схема алгоритму пошуку функцій

Кожне число відповідає певній функції. Тому число переводиться у масив довжиною *n*. Кожен елемент масиву відповідає розряду регістра за таким же номером по порядку. Значення елементів масиву відповідають певній логічній операції з розрядом регістру. В залежності від значення можливі варіанти, як – розряд не є операндом функції затримки (значення 0), розряд входить до функції та з ним виконується операція І (значення 1), АБО (значення 2), ВИКЛЮЧНЕ АБО (значення 3) та інші. Далі, після отримання сформованої функції, перевіряється чи всі операнди та логічні операції є

дозволеними користувачем. І при умові, що всі задовольняють умовам відбувається перевірка функції в ролі функції затримки регістра зсуву.

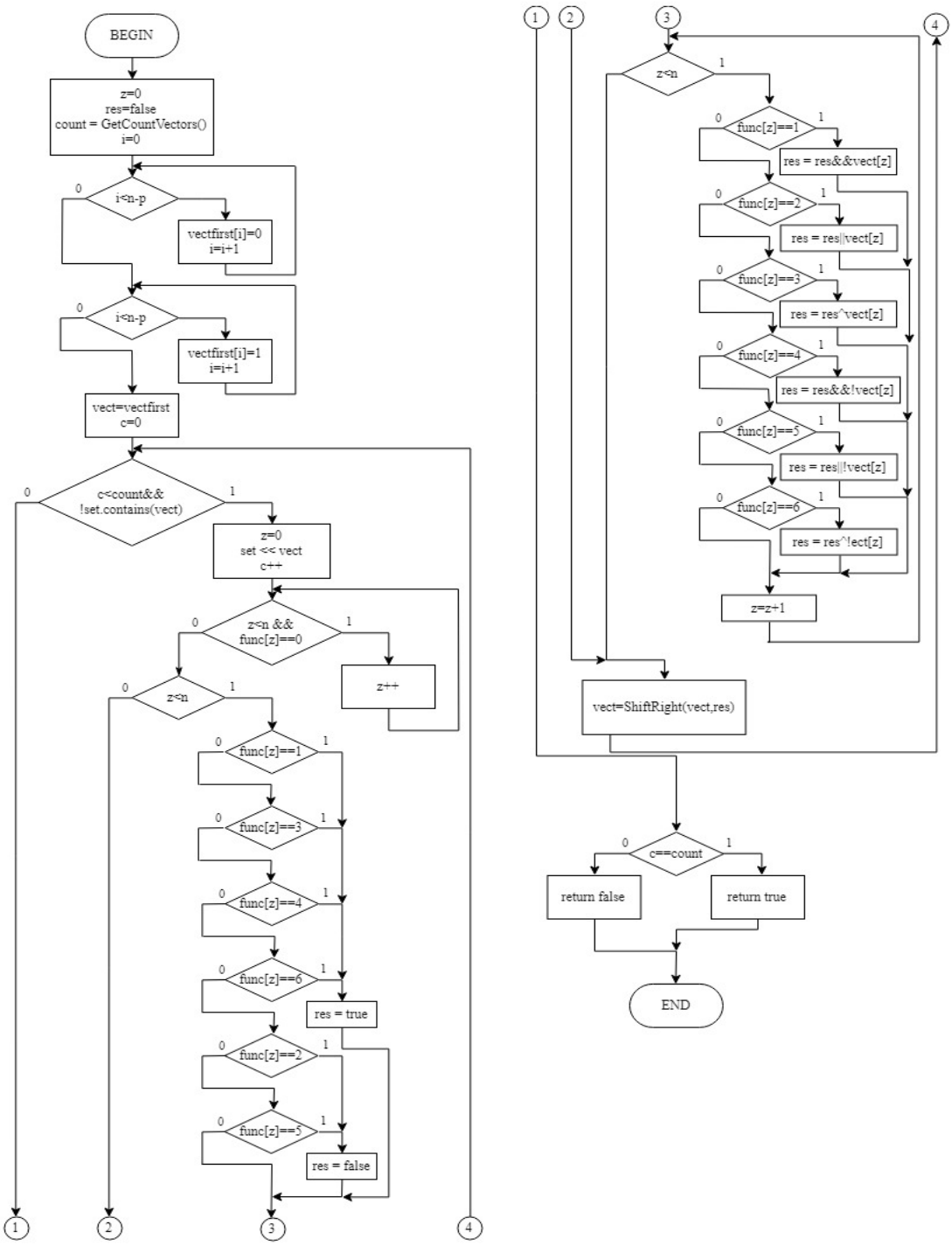

Рисунок 8 – Алгоритм перевірки функції управління

Тобто за допомогою функції CheckFunc(f), що на рис.8, перевіряється утворений масив значень ФУ на формування послідовності зі всіх можливих

векторів заданих параметрів без повторень. Після цього формується наступне число, яке задаватиме нову функцію управління для перевірки.

Перевірка функції відбувається в методі CheckFunc(f), який враховуючи параметри пошуку, введені користувачем, перевірить сформовану послідовність векторів. Чи буде послідовність містити всі вектори заданої ваги без повторів. Послідовність роботи видно на рис. 8, де відображений алгоритм дій. Спочатку виконується розрахунок кількості векторів заданої ваги при вказаній довжині. Це дає можливість слідкувати за періодом отриманої групи векторів та перевіряти чи є максимальним. Далі в залежності від ваги створюється масив бітів, що містить в собі p одиниць і n-p нулів. Так як результуюча послідовність шуканої функції має бути максимального періоду, то початковий вектор може бути будь-яким з групи векторів вказаних параметрів.

Після того, як сформований перший вектор послідовності, він додається до множини всіх отриманих векторів і лічильник збільшується на 1. Далі відбувається прохід по всім елементам масиву функції і відповідно до значень елементів визначається логічна операція для кожного біту регістра чи її відсутність. Таким чином спочатку буде отримане початкове значення для розрахунку функції, а далі вже обчислення результату. Отримане значення впливає на те, чи зсув відбуватиметься з затримкою, чи без. І зважаючи на це, обчислюється наступний вектор послідовності. Якщо такого вектора ще не було, а кількість вже сформований менша за потрібний період, то визначається наступний стан генератора. Це повторюється поки послідовність не зациклиться або ж не будуть сформовані усі вектори послідовності.

Якщо після виходу з циклу кількість утворених векторів за допомогою вказаного генератора й функції буде меншою за потрібну кількість, то функція не може використовуватись у ролі ФУ генератором визначеної конфігурації. Якщо ж умови виконані, то функція є дійсною і додається до результуючого масиву отриманих функцій.

Всі ці кроки повторюються для всіх можливих значень функцій довжини *n* і виключаються тільки такі, що не є дозволеними користувачем або ж не можуть виконувати роль функцій затримки регістра. Після того, як масив ФУ уже сформований, то він повертається як результат роботи функції.

3.4. Формування послідовності двійкових рівновагових векторів

Клас Register відтворює результати роботи регістра зсуву, що дозволяє задати початкові значення бітів у регістрі, задати ФУ та отримати сформовану послідовність РДВ. Також методи цього класу дають можливість форматувати вивід векторів, перевіряти чи підходять БФ для поставленої задачі, отримати масив векторів, який є послідовністю РДВ, згенерованою з умовою ФУ.

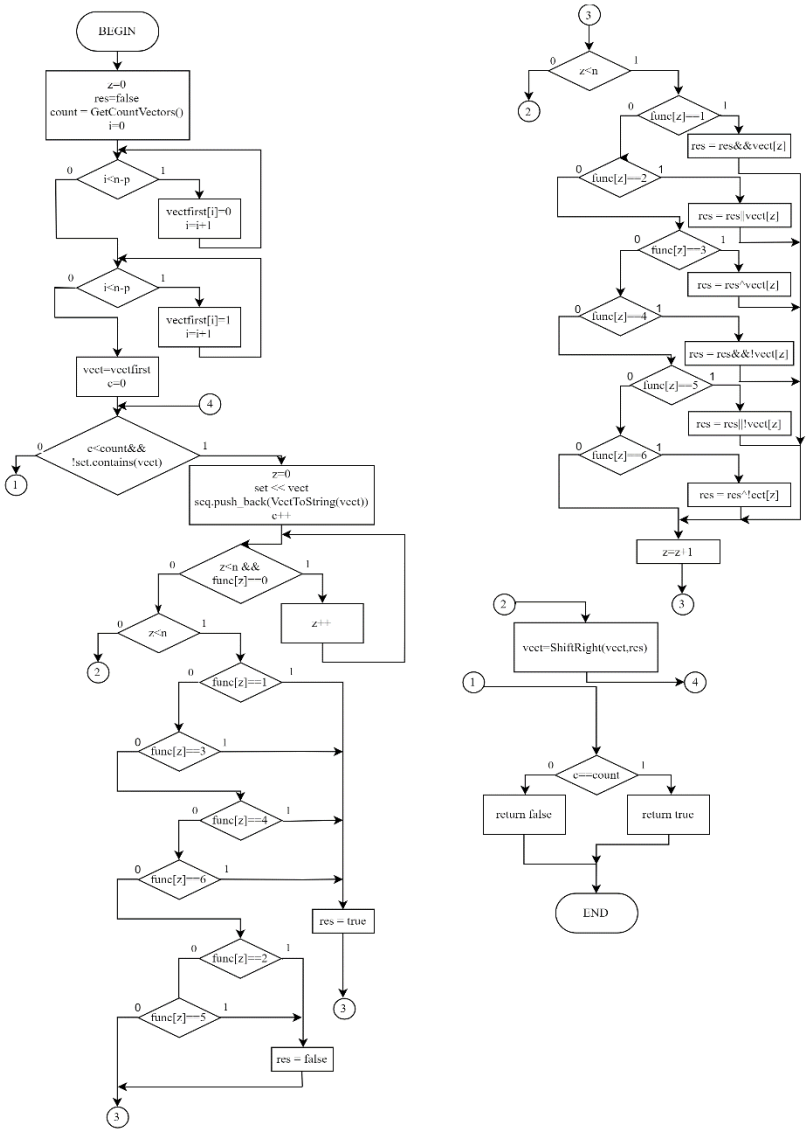

Рисунок 9 – Алгоритм утворення послідовності РДВ

Послідовність РДВ довжини *n* і ваги *p* відбувається під час виклику функції GetSequence() об’єкта класу Register. Схема алгоритму відображена на рис. 9. В даному методі спочатку утворюється початковий вектор. Для цього відбувається *p* ітерацій циклу, що записує одиниці та *n-p* ітерацій циклу запису нулів. Так отримується перший вектор, якій має вказані користувачем довжину та вагу.

Далі створюється пуста множина всіх вже сформованих векторів, куди додається перший. А також лічильник векторів стає рівним одиниці. Ще сформований вектор викликаючи функцію VectToString(vect) переводить його у рядок та додає до результуючого масиву елементів послідовності.

В методі VectToString(vect) створюється пустий рядок, який міститиме вектор у вигляді символів. Виконується прохід по масиву вектора за допомогою циклу з лічильником, де числове значення елемента переводиться у текстове та додається до рядку.

Виконується підрахунок значення функції на даному масиві бітів і в залежності від результату, отримуємо наступний вектор послідовності. При значенні 0 – відбувається зсув вправо без участі першого розряду, а якщо 1 – то перший бід масиву залишається на місці, а всі інші зсуваються вправо і останній біт стає другим елементом масиву.

Таким чином виконується формування послідовності, поки не будуть утворені всі РДВ, тобто буде зациклення. Отримана послідовність буде містити всі вектори заданої довжини, з однаковою вагою. Метод поверне масив рядків, кожен з яких є текстовим представленням РДВ.

3.5. Перевірка працездатності створеного програмного забезпечення

Написана програма має зчитувати два введенні користувачем числа, які більше або рівні одиниці та вказана вага має бути меншою за кількість бітів регістра. Якщо користувач не введе дані, то це контролюється і програма видасть повідомлення, що продемонстровано на рис. 10.

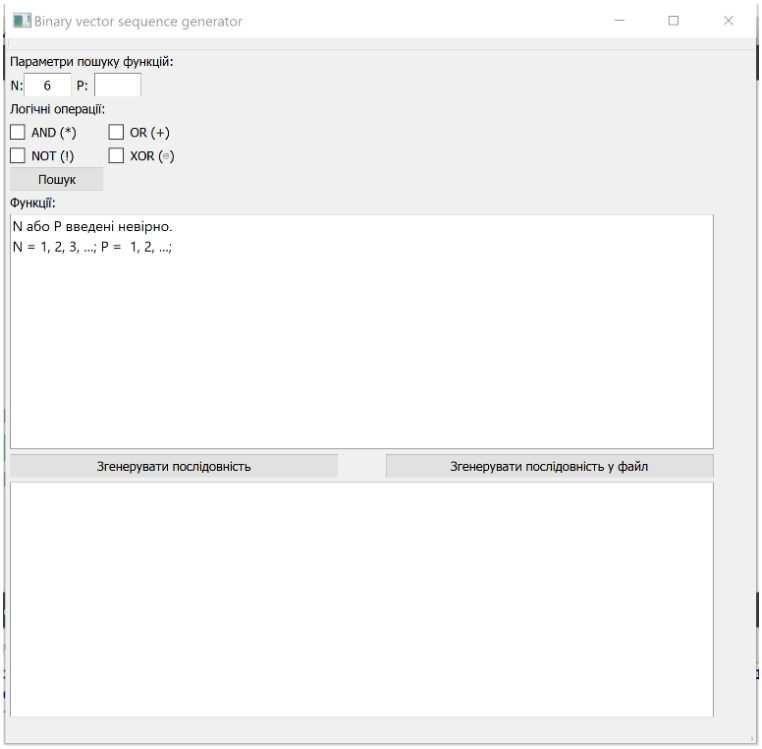

Рисунок 10 – Реакція програми на відсутність вхідним даних

Якщо попередні умови не виконуються, то при натисненні кнопки генерування послідовності буде видане повідомлення, що введені числа не є коректними. Перевірка таких дій відображена на рис. 11.

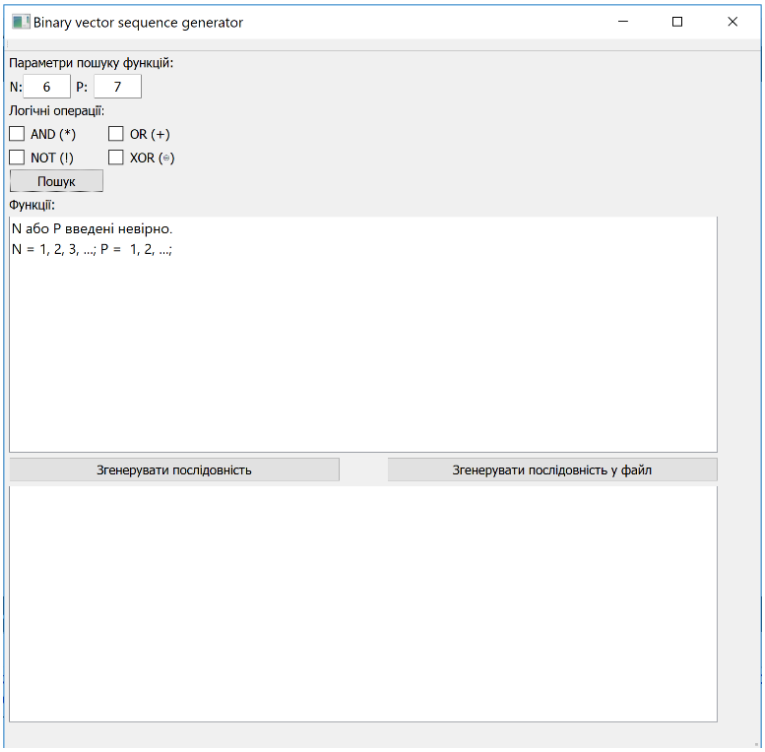

Рисунок 11 – Реакція програми на некоректні умови

Після вірно введених параметрів пошуку, користувач має обрати логічні операції. Це вказуватиме на те, які з них дозволенні у ФУ. Якщо жодна з операцій не є обраною, то програма вимагатиме вибору хоча б однієї зі всіх, що можна спостерігати на рис. 12.

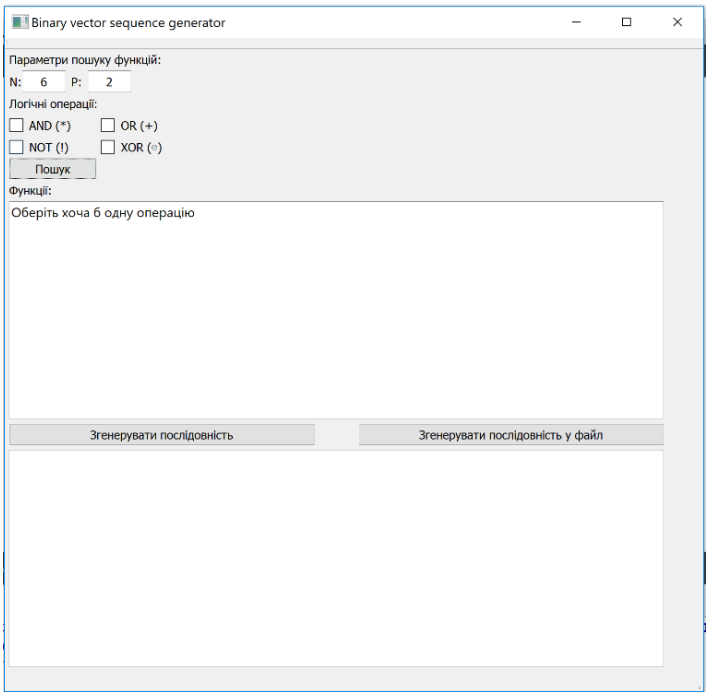

Рисунок 12 – Повідомлення про відсутність вибору операції

Після всіх вірно введених даних програма знайде всі ФУ для даного регістра зсуву і виведе їх, з можливістю вибору. Варто зауважити, що поки кнопка «Пошук» знаходиться у активному стані (в той час її колір є блакитним), програма виконує пошук і перевірку ФУ, тому не відповідатиме на інші дії користувача. Після попередніх кроків користувач має обрати одну ФУ для подальшої роботи програми. Після натиснення кнопок генерування послідовностей, в налаштування регістра запишеться обрана ФУ. Якщо ж користувач не обере жодної, то буде видане повідомлення, що підтверджує коректну роботу програми. Вибір функції та результат генерування послідовності видно на рис. 13. Видно, що всі етапи пошуку, генерування виконані й результат також був збережений до файлу, який надалі може використовуватись поза межами цього ПЗ. Також можна порахувати, що

отримано 20 різних РДВ у послідовності, що підтверджує правильність роботи описаного генератора.

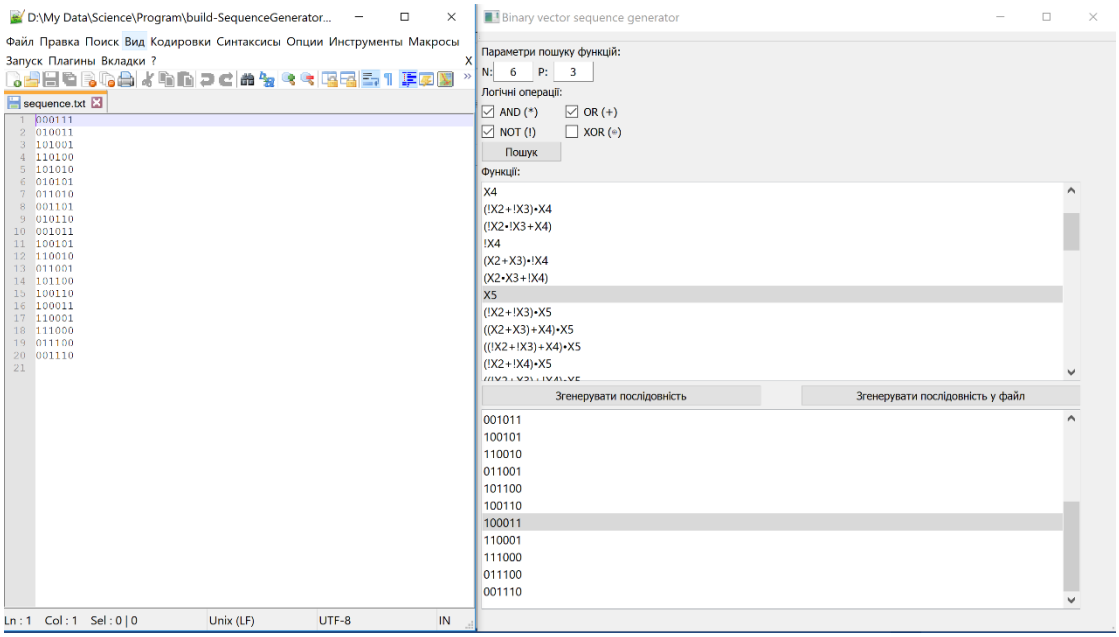

Рисунок 13 – Відображення послідовності РДВ на екрані та у файлі

Беручи до уваги всі виконанні етапи тестування, перевірки працездатності програми, можна стверджувати що робота є коректною, умови виконані, результати є вірними, а також контролюються користувацькі помилки.

ВИСНОВКИ

В даній роботі розглянуто існуючі методи формування псевдовипадкових послідовностей двійкових векторів однакової ваги, який побудовано на базі регістру зсуву. Розроблено алгоритми формування функцій управління окремими розрядами регістру зсуву, їх пошуку та генерації послідовності. Генератор побудований на базі регістру зсуву з прямим зворотнім зв'язком. Особливістю є управління першим розрядом під час зсуву вправо, що визначається значенням функції управління генератором.

Розроблена програма дозволяє:

- виконувати пошук функцій управління генератором;
- задавати параметри пошуку, такі як довжина регістру, вага векторів і допустимі логічні операції;
- формувати послідовність двійкових рівновагових векторів використовуючи обрану користувачем функцію;
- записати сформовану послідовність до текстового файлу в пам'яті комп'ютера.

В подальшому функціональність програми може бути розширена за допомогою додавання можливості управління не тільки першим розрядом регістра під час зсуву.

Головною особливістю побудованого генератора є безповторність всіх можливих рівновагових векторів протягом періоду. Генератор виконує роль джерела векторів стану процесорів багатопроцесорної системи під час виконання статистичного експерименту з моделями поведінки відмовостійких багатопроцесорних систем у потоці відмов у процесі розрахунку їх надійності.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Методи генерації випадкових чисел з рівномірним законом розподілу [Електронний ресурс] - https://habr.com/ru/post/301900/
2. Генератори випадкових послідовностей [Електронний ресурс] - https://habr.com/ru/post/151187/
3. Моделювання систем. Генеартори випадкових чисел [Електронний ресурс] - http://stratum.ac.ru/education/textbooks/modelir/
4. Інформатика і кібернетика. Генератори випадкових і псевдовипадкових чисел. [Електронний ресурс] - https://moluch.ru/conf/tech/archive/286/13233/
5. Rabah AlShboul, Vitaliy A. Romankevich. Method of Numbers' Dichotomic Decomposition for Generation of Equal Probability Binary Sets // IJCSNS International Journal of Computer Science and Network Security.– 2019.– Vol. 19, No.2.– P. 120-125.
6. Моделирование. Тестирование, надёжность, контроль и диагностика компьютерных систем [Електронний ресурс]: учебное пособие для изучения дисциплин «Моделирование» и «Тестирование, надёжность, контроль и диагностика компьютерных систем» для иностранных студентов специальности «Компьютерные системы и сети», «Системное программирование» и «Специализированные компьютерные системы» / НТУУ «КПІ» ; сост. В. В. Гроль, В. А. Романкевич, Е. Р. Потапова.– Електронні текстові дані (1 файл: 782.5 Кбайт). – Киев: НТУУ «КПИ», 2011
7. Романкевич В.О., Фесеннюк А.П. Про розрахунок надійності відмовостійких багатопроцесорних систем, підсистеми яких мають спільні процесори // Радіоелектронні і комп'ютерні системи.– №3, 2010.– №3.– С. 62-67
8. Романкевич В.О., Майданюк І.В., Фесенюк А.П., Шкира Д.С. Генератор рівноважних векторів для проведення статистичних експериментів з GL-

моделями // Науковий вісник Чернівецького університету. Сер.: Комп'ютерні системи та компоненти.– 2010.– Т.1, вип. 2.– С.28-30

9. Rabah AlShboul, Vitaliy A. Romankevich. Structural Means Generating Pseudorandom Sequences Of Fixed Weight Binary Patterns // IJCSNS International Journal of Computer Science and Network Security. – 2017. – Vol. 17, No.10. – P. 62-66.
10. Романкевич В.А., Майданюк И.В. Структурный метод формирования двоичных псевдослучайных векторов заданного веса // УСиМ. - 2011. - № 5. - С. 28-33, 58.
11. В.В. Гроль, В.А. Романкевич, А.П. Фесенюк. Об оценке погрешности расчета надежности отказоустойчивых многопроцессорных систем // Радіоелектронні і комп'ютерні системи.–№5, 2009.– С.56-59
12. Гроль В.В., Романкевич В.А., Потапова Е.Р., Мораведж Сейед Милад. Структурный метод генерации псевдослучайных последовательностей специального вида // Радіоелектронні і комп'ютерні системи.–№5, 2010.– С.230-236
13. Гроль В.В., Романкевич В.О. Базові поняття і конструкції мови програмування Сі. Методичні вказівки до вивчення дисципліни "Моделювання" // Київ.– "Політехніка", 2003.– 24с.
14. Документація бібліотеки Qt [Електронний ресурс] - http://qt-doc.ru
15. Самофалов К.Г., Романкевич А.М., Валуйский В.Н., Каневский Ю. С. , Пиневич М.М. Прикладная теория цифровых автоматов. – К.: Вища Школа. – 1987г. – 375с.
16. Романкевич А.М., Майданюк И.В., Романкевич В.А О формировании функций управления для генератора последовательностей двоичных векторов // Радіоелектронні і комп'ютерні системи.-№6, 2014.- С.157-163
17. Романкевич О.М., Карачун Л.Ф., Романкевич В.О. До питання побудови моделі поведінки багатомодульних систем // Наукові вісті НТУУ "КПІ".– 1998.– №1.–С.38-40.

18. Гроль В.В., Романкевич В.А., Потапова Е.Р. Организация процедур логического моделирования цифровых блоков. Методические указания к изучению дисциплин «Моделирование», «Тестирование, надёжность, контроль и диагностика компьютерных систем» // Київ.– “Принт-центр”, 2007.– 44с.
19. Романкевич А.М, Гроль В.В., Карачун Л.Ф., Орлова М.Н., Романкевич В.А. Об одном подходе к расчёту надёжности отказоустойчивых многопроцессорных систем // ВМНТС "АСУ и приборы автоматики".– ХНУРЭ, Харьков.– 2002.– №119.–С.54-58.